UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA UNIVERSITARIA

DE INGENIERÍA TÉCNICA

DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

APLICACIÓN MÓVIL PARA EL ACCESO A JUEGO DE NAIPES SOBRE EL DESARROLLO HUMANO

Autor

Miguel Ángel Gilpérez Alcázar

Tutor

Miguel Ángel Valero

Septiembre de 2013

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Plataforma cooperativa para juego de naipes sobre indicadores de desarrollo humano

TÍTULO: Aplicación móvil para el acceso a juego de naipes sobre el desarrollo humano

AUTOR: Miguel Ángel Gilpérez Alcázar

TUTOR: Miguel Ángel Valero Duboy **Vº Bº.**

DEPARTAMENTO: Dpto. de Ingeniería y Arquitecturas Telemáticas

Miembros del Tribunal Calificador:

PRESIDENTE: Manuel Vázquez Rodríguez

VOCAL: Miguel Ángel Valero Duboy

VOCAL SECRETARIO: Carlos Ramos Nespereira

Fecha de lectura: 27 de septiembre de 2013

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

Este Proyecto Fin de Carrera (PFC) tiene como objetivo el análisis, diseño e implementación de un videojuego móvil multijugador, con un enfoque educativo, para la sensibilización sobre el Índice de Desarrollo Humano (IDH). El sistema resultante se ha desarrollado para la Plataforma Android, utilizando el Framework AndEngine, de modo que pueda utilizarse en un amplio número de terminales móviles disponibles en el mercado.

La aplicación se presenta como un juego de cartas con los diferentes países y sus datos, en donde los jugadores deben conocer el peso de los índices de desarrollo (esperanza de vida, renta, educación) de los países en comparación con los países de los otros jugadores. La actualización de la información y de los datos de las partidas se realiza a través de la comunicación con un servidor web ya implementado de forma complementaria a la realización de este proyecto.

El sistema ha sido integrado y validado satisfactoriamente con diferentes terminales móviles y usuarios de diferente perfil de edad y uso.

Resumen

Este Proyecto Fin de Carrera (PFC) tiene como objetivo el análisis, diseño e implementación de un videojuego móvil multijugador, con un enfoque educativo, para la sensibilización sobre el Índice de Desarrollo Humano (IDH).

El sistema resultante se ha desarrollado para la Plataforma Android, utilizando el Framework AndEngine, que utiliza aceleración hardware de la GPU para garantizar un buen rendimiento en terminales de gama baja, de modo que pueda utilizarse en un amplio número de terminales móviles disponibles en el mercado.

La aplicación se presenta como un juego de cartas con los diferentes países y sus datos humanitarios, los jugadores deben conocer el peso de los índices de desarrollo (esperanza de vida, renta, educación) de los países en comparación con los países de los otros jugadores. El sistema de juego premia a los jugadores con mayores conocimientos sobre los datos humanos de los diferentes países del mundo, de ese modo los mejores jugadores serán los que tengan más conocimientos de estos datos.

El juego permite jugar partidas en solitario utilizando jugadores manejados por la CPU, o multijugador mediante WIFI o 3G.

La actualización de la información y de los datos de las partidas se realiza a través de la comunicación con un servidor web ya implementado de forma complementaria a la realización de este proyecto.

El sistema ha sido integrado y validado satisfactoriamente con diferentes terminales móviles y usuarios de diferente perfil de edad y uso.

El videojuego se puede descargar de la página web creada en un proyecto complementario a este (pendiente de publicación web), y ya se encuentra también disponible en Google Play.

https://play.google.com/store/apps/details?id=xnetcom.pro.cartas&hl=es_419

Resumen

This Project End of Career (PFC) takes as an aim the analysis, design and implementation of a multiplayer mobile videogame, with an educational approach, for the awareness on the Human Development Index (HDI).

The resultant system has been developed for the Platform Android, using the AndEngine Framework, which uses hardware acceleration of the GPU to ensure a good performance on low-end terminals, so that it can be used in a wide range of mobile handsets available in the market .

The application is presented as a card game with the different countries and his humanitarian information, the players must know the weight of the indexes of development (life expectancy, revenue, education) of the countries in comparison with the countries of other players.

The game system rewards players with more knowledge on human information of different countries, thus the best players will be those with more knowledge of these information.

The game allows to play items in solitarily using players handled by the CPU, or multiplayer by means of WIFI or 3G.

The update of the information and data of the online games is done through communication with a web server implemented as a complement to the realization of this project.

The system has been built and successfully validated with different mobile terminals and users of different age and usage profile.

The game can be downloaded from the website created in a complementary project to this (web publication pending), and is now also available on Google Play

https://play.google.com/store/apps/details?id=xnetcom.pro.cartas&hl=es_419

AGRADECIMIENTOS

Quiero dar las gracias a todos los que han me han acompañado a lo largo de este viaje.

A mis padres, por su apoyo y confianza en los buenos y malos momentos, sin ellos nada de esto habría sido posible.

A todos los compañeros y amigos de la universidad, por su amistad y compañerismo y por todos esos momentos inolvidables. Especialmente a Merche, porque sin sus apuntes posiblemente no estaría escribiendo estas líneas.

A todos esos buenos profesores que he tenido a lo largo de la carrera y especialmente a Miguel Ángel por ofrecerme este proyecto del que he disfrutado y aprendido tanto.

ÍNDICE

ÍNDICE DE FIGURAS

1 PRESENTACIÓN

1.1 MOTIVACIÓN

La motivación principal de este Proyecto Fin de Carrera (PFC) es la divulgación del Índice de Desarrollo Humano (IDH) con objeto de contribuir a la concienciación de los usuarios sobre las desigualdades que existen entre los países y continentes, a través del conocimiento de las diferencias en los indicadores de los países más ricos y más pobres, de la forma más educativa posible.

Considerando el despliegue de las tecnologías de la información y comunicación actuales, se ha optado por la propuesta de diseñar y desarrollar un videojuego multijugador online, con un servidor y un portal que se ha realizado en un PFC complementario titulado "Servicio Web para gestión de juego cooperativo sobre desarrollo humano", cuya autora es Mercedes Sánchez Maroto. La realización de este PFC para una plataforma móvil se debe a la gran importancia y penetración que tiene actualmente en el mundo la telefonía móvil en el panorama actual. Los smartphones y tablets son dispositivos relativamente nuevos, pero hoy en día son equipos considerados por muchos casi como imprescindibles y gran parte de ese impacto proviene del mercado de aplicaciones disponibles.

Este PFC también contribuye a la consolidación y adquisición de conocimiento sobre tecnologías utilizadas durante la carrera, tanto para el desarrollo de aplicaciones móviles en Java y Android, como en XML y el modelo Cliente-Servidor.

Desde el principio se diseñó como una aplicación multijugador para que pudiera acercar a personas del mundo entero que estuvieran interesadas en el IDH. Así, además de jugar, pueden intercambiarse opiniones y puntos de vista sobre los índices de otros países o los de sus respectivos países con personas de cualquier parte del mundo. La interconexión del sistema, junto con la web y el foro implementado en el PFC del servidor complementario ofrece a los usuarios una plataforma didáctica y lúdica centrada en el IDH.

1.2 OBJETIVOS

La misión de este Proyecto Fin de Carrera (PFC) es contribuir al conocimiento del Índice de Desarrollo Humano (IDH), de forma lúdica, a través del teléfono móvil. En consecuencia, los usuarios podrán ser más conscientes de la situación real del mundo y de las desigualdades que existen en él. El IDH persigue así mismo que la población, en nuestro caso los usuarios del PFC, no juzguen a un país exclusivamente por su Producto Interior Bruto (PIB). Con este fin, se propone el siguiente objetivo:

> "Desarrollar y validar un aplicación para dispositivos móviles Android que, desde un enfoque educativo y lúdico, favorezca el conocimiento sobre el Índice de Desarrollo Humano de los países."

La consecución del objetivo principal anteriormente enunciado requiere cubrir los siguientes objetivos específicos:

- Conocer el Índice de Desarrollo Humano y sus variables principales
- Estudiar las capacidades de Android para el desarrollo de juegos multimedia multijugador
- Especificar las características de la aplicación lúdica y sus reglas de juego mediante un análisis detallado en UML

1.3 ORGANIZACIÓN DE LA MEMORIA

El presente documento se ha estructurado en siete capítulos y dos anexos. En el Capítulo 1, Presentación, se describen los objetivos y la motivación que han llevado a la creación de este FPC. Los antecedentes sobre el IDH, la situación actual de la tecnología utilizada y un estudio sobre los videojuegos se recogen en el Capítulo 2.

En los capítulos 3 y 4 se incluye, respectivamente el Análisis del Sistema y Diseño de la aplicación. En primer lugar se explica la funcionalidad y diagramas UML del sistema y posteriormente se detalla el diseño del protocolo de transmisión entre el cliente y el servidor, la interfaz de usuario gráfica y el diseño del diagrama de clases previo a la implementación. El Capítulo 5 recoge la Implementación detallas y Pruebas técnicas y de usuario realizadas.

Finalmente el Capítulo 6 incluye la Conclusiones y futuras mejoras que se han obtenido tras la realización de este PFC. El Capítulo 7 recoge la Bibliografía empleada y a continuación los Anexos A y B con el Glosario de acrónimos y código de la aplicación.

2 ANTECEDENTES

En el presente capítulo se detallan las bases conceptuales y tecnológicas que han servido de fundamentación para la elaboración de este Proyecto Fin de Carrera.

2.1 EL ÍNDICE DE DESARROLLO HUMANO

El Índice de Desarrollo Humano (IDH) fue creado por el Programa de las Naciones Unidas para el Desarrollo (PNUD) [1], con la finalidad no sólo de contar con los ingresos económicos de las personas de un país, sino la situación de la vida humana para tener una visión más amplia. Los Informes sobre Desarrollo Humano se basa en un índice que trata de medir las condiciones de la de la vida humana, basándose en un indicador social estadístico compuesto por tres parámetros: salud, educación y riqueza. Desde 1990 estos documentos son actualizados cada año por un equipo de consultores independientes para el PNUD con el fin de mantener el informe intelectual y editorialmente independiente. Los informes constituyen una herramienta muy útil a la hora de determinar qué aspectos del desarrollo humano necesitan más atención en las diferentes partes del mundo.

A raíz de la realización de estos informes, aunque a menudo provocadores pero siempre con una sólida base de investigaciones empíricas, algunos gobiernos han solicitado la ayuda del PNUD para priorizar las ayudas a las necesidades del desarrollo humano en sus países, actualmente unos 30 países están siendo asistidos en la formulación e implementación de estrategias presupuestarias en beneficio de las personas.

Desde su creación los Informes han planteado ideas pioneras sobre en los conceptos de desarrollo humano y han ofrecido gracias a su riqueza de datos, nuevos enfoques para medir el desarrollo humano, gracias a esto han tenido una gran acogida por muchos gobiernos. Estas ideas han planteado debates gracias a los cuales se han generalizado a nivel mundial las bases de conceptos como los Objetivos de Desarrollo de Milenio (ODM) [2].

El enfoque del desarrollo humano se basa en la idea del desarrollo que fue planteada por primera vez por Mahbub Ul Haq, principal autor de los Informes sobre Desarrollo Humano iníciales, y Amartya Sen, Premio Nobel en Economía, quien asesoró y ayudó a Haq a la hora de desarrollar el enfoque filosófico general del informe.

El Informe de 1990 afirmó que el objetivo del desarrollo humano es "ampliar las oportunidades de las personas", incluyendo la capacidad de disfrutar de vidas saludables, recibir formación y contar con un nivel de vida aceptable. Los primeros informes también hicieron hincapié en que el desarrollo humano y el bienestar van más allá de estas dimensiones, integrando un mayor rango de capacidades, incluyendo las libertades políticas y los derechos humanos. De hecho, el "concepto" de desarrollo humano es mucho más amplio que la "medida" del desarrollo humano en sí (por ejemplo, mediante el IDH). El buen recibimiento que tuvo entre gobiernos, sociedad civil, investigadores y medios de comunicación demostró la enorme eco que tuvo este innovador enfoque, no sólo entre la comunidad de desarrollo, sino en la sociedad en general [3].

Los informes de cada año han enfatizado un tema por ejemplo en el informe de 1991, que versaba sobre desarrollo económico y avisaba sobre los daños en la capa de ozono, el concepto de seguridad humana, tema del Informe de 1994, recalcó la importancia de la sostenibilidad para el enfoque de desarrollo humano. El Informe de 1999 destacó los diversos efectos que tiene la globalización sobre el bienestar humano. El Informe 2011, aprovechando este legado, identificó las políticas que pueden aportar simultáneamente sostenibilidad y equidad, tanto local como globalmente. Pero todas ellas hacen siempre hincapié en la prioridad por el bienestar de las personas.

El *objetivo último del Informe* es ayudar a avanzar en el desarrollo humano, lo que supone prestar una especial atención a la salud, la educación y la ampliación de las libertades y habilidades humanas. La frase inicial del primer Informe sobre Desarrollo Humano de 1990 afirmaba que "la verdadera riqueza de una nación son sus personas". Es importante reseñar a este respecto la importancia de la independencia y objetividad de los informes tal como enfatiza la cita del informe de 2002: "El Informe depende de las estadísticas obtenidas a partir de una amplia gama de agencias multilaterales y de Naciones Unidas, pero su análisis y las conclusiones son responsabilidad exclusiva de sus autores. Su autonomía editorial está protegida por una resolución de la Asamblea General (A/RES/57/264), que reconoce que el Informe sobre Desarrollo Humano es "un ejercicio intelectual independiente" y "una herramienta importante de concienciación sobre el desarrollo humano en todo el mundo".

El IDH se calcula a partir de la elaboración de dimensiones fundamentales básicas: la salud, la educación y la riqueza de los habitantes de un país [4].

A continuación se detallan resumidamente los tres parámetros básicos en los que se sostiene el IDH.

1. **Salud:** Es el promedio de edad de las personas fallecidas al año. se ha seleccionado por la creencia universal del deseo de tener una vida prolongada y estrechamente relacionada con aspectos que mantienen la calidad de vida.
2. **Educación:** Recoge el nivel medio de alfabetización adulta y el promedio de años de escolaridad, refleja la importancia que se le da a la formación y lo que significa para el desarrollo humano.
3. **Riqueza:** Recoge diversos factores como el producto interior bruto PIB. El PIB Indica la posibilidad de las personas de adquirir bienes y servicios a fin de satisfacer sus necesidades, indica en qué medida las personas pueden tener acceso a recursos para poder obtener un nivel de vida adecuado.

Los valores de estos parámetros están comprendidos entre valores de "0" y "1" siendo "0" la puntuación más baja y "1" la más alta, según su puntuación global los países quedan clasificados en cuatro grandes grupos:

I. ***Países con muy alto desarrollo Humano:*** ("High Human Development") Tienen un IDH mayor de 0.80.
II. ***Países con alto desarrollo Humano:*** ("High Human Development") Tienen in IDH entre 0,70 y 0.80.
III. ***Países con medio desarrollo Humano:***("Medium Human Development") Tienen in IDH entre 0.50 y 0,70.
IV. ***Países con bajo desarrollo Humano****:* ("Low Human Development") Tienen un IDH menor de 0.50.

A continuación se muestra un Mapa del mundo según su IDH.

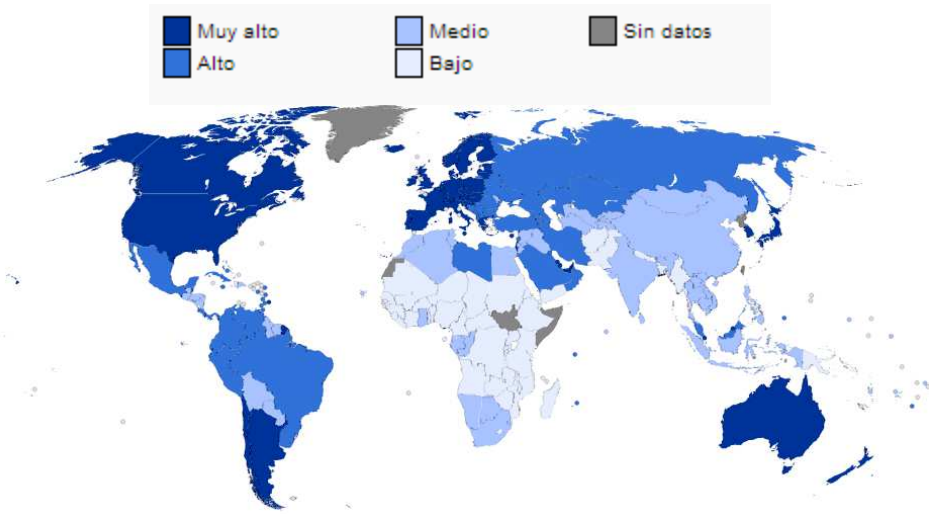


Ilustración 1. Mapa del Mundo extraído del Informe sobre Desarrollo Humano 2011.

En la tabla siguiente puede verse la evolución histórica entre 1980 y 2012 del IDH de los países situados en las primeras posiciones, añadiendo España, EEUU y Alemania.

Países	1980	1990	2000	2005	2006	2007	2012
Noruega	1	2	1	1	1	1	1
Australia	11	12	3	2	2	2	2
Islandia	7	7	10	3	3	3	13
Canadá	4	1	6	4	4	4	11
Irlanda	21	20	15	5	5	5	7
Estados Unidos	3	3	5	12	12	13	**3**
España	19	15	17	15	15	15	23
Alemania	13	16	N/A	21	22	22	5

El informe de 2013 cuenta con 186 países frente a los 169 que se incluían en el informe de 2010 [5]. Este aumento se debe a la colaboración de la oficina encargada del Informe sobre Desarrollo Humano y a los proveedores internacionales de datos y agencias nacionales de estadística. Este número se ha ido incrementando desde su creación, si bien ha variado en los informes debido principalmente a la falta de datos en algunos países. En el informe de 2013 no se pudo incluir a estados miembros como Islas Marshall, Mónaco, Nauru, República Popular Democrática de Corea, San Marino, Somalia, Sudán del Sur y Tuvalu.

IDH ajustado por la Desigualdad (IDH-D).

Según el PNUD, el Índice de Desarrollo Humano ajustado por la Desigualdad (IDH-D) adapta el Índice de Desarrollo Humano (IDH) en base a la desigualdad de distribución en cada dimensión existente entre toda la población. El IDH-D recoge las desigualdades en las dimensiones del IDH "rebajando" la media del valor de cada dimensión según su nivel de desigualdad. El IDH-D es igual al IDH cuando no existen desigualdades entre las personas, pero será inferior al IDH en la medida que crezca dicha desigualdad. En este sentido, el IDH-D es el nivel real de desarrollo humano (teniendo en cuenta esta desigualdad), mientras que el IDH puede considerarse el índice de desarrollo humano "potencial" (o el nivel máximo de IDH) que puede lograrse en caso de que no existan desigualdades. La "pérdida" de potencial de desarrollo humano debida a la desigualdad queda reflejada en la diferencia existente entre el IDH y el IDH-D, y puede expresarse en forma de porcentaje.

El Índice de Pobreza Multidimensional.

Según el PNUD, El Índice de Pobreza Multidimensional (IPM) identifica las diversas privaciones a nivel individual en salud, educación y nivel de vida. Utiliza los microdatos de las encuestas familiares y, al contrario que el Índice de Desarrollo Humano ajustado por la Desigualdad, todos los indicadores necesarios para elaborar la medida deben provenir de la misma encuesta. Cada miembro de una misma familia se clasifica como pobre o no pobre, dependiendo del número de privaciones que sufra su familia. Estos datos se agregan entonces a las medidas nacionales de pobreza.

El Índice de Desigualdad de Género (IDG).

Este índice trata de cuantificar la diferencias en el desarrollo entre mujeres y hombres en tres dimensiones, salud reproductiva, empoderamiento y mercado laboral, sus valores están comprendidos entre el 0 que significa que mujeres y varones presentan un desarrollo igual, y el 1, que supone que las mujeres tienen el peor desarrollo posible en todas las dimensiones medidas.

2.2 VIDEOJUEGOS

La Real Academia Española define videojuego como un "dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador". En lo que respecta a este PFC, se considera videojuego educativo aquel cuya meta sea la obtención de algún tipo de conocimiento por parte de los usuarios, ya sea, por ejemplo, en geografía, historia o matemáticas, entre otras disciplinas. Este tipo de juegos suele estar enfocado al público más joven, por lo que su contenido y modos de interacción deben de ser adecuados al público al que están dirigidos.

2.2.1 Breve historia de los videojuegos

Los videojuegos aparecieron a finales de los años 50, utilizaban osciloscopios y circuitos analógicos para representar los objetos y no existían gráficos. Más tarde, en los años 70 aparecieron los gráficos vectoriales y por fin las videoconsolas que podían reproducir imágenes de bitmap en pantalla, siendo el Galaxy Game el primer videojuego en utilizar esta tecnología y considerado por muchos como el primer videojuego. Más tarde saldrían al mercado juegos de gran éxito como el Pong y el Space Invaders, quienes darían un gran empujón a la industria de los videojuegos.

La industria de los videojuegos siguió creciendo a gran velocidad en los años 80, comenzaron a nacer gran cantidad de empresas muchas de ellas en Japón, como Nintendo o Namco, creadora esta última del videojuego Pac-Man, que consiguió en su momento el record Guines del videojuego de más exitoso de todos los tiempos con un total de 293.822 máquinas vendidas entre 1981 y 1987. Nintendo por su parte lanzo la plataforma NES, (Nintendo Entertainment System) que se convirtió rápidamente en un gran éxito con unas ventas totales de 61.9 millones de consolas según Nintendo, a lo largo de su producción, esta plataforma estaba apoyada por juegos de la talla de Donkey Kong , Mario Bros o Zelda. En la década de los 90 la evolución de los juegos siguió su curso lógico, con juegos de mayor calidad gráfica y en 3D que se convertiría en la forma dominante de creación de juegos en el futuro. La salida de la consola de Sony la PlayStation en 1994 de la que se llegaron a vender más de 100 millones, supuso el estallido de los videojuegos en 3D.

La década de los 2000 supuso un cambio en la forma de jugar, Nintendo con su Wii revoluciono la forma en la que el usuario se comunicaba con el sistema, en ella el movimiento del jugador sustituía a la anterior forma de comunicación mediante la pulsación de botones. Esta década se caracterizó también por la explosión de los juegos en red, teniendo como máximo representante a World of Warcraft.

2.2.2 Tipos de videojuegos

El mercado de los videojuegos está en constante ebullición y casi cada día hay novedades en el mercado, los desarrolladores buscan siempre ofrecer algo nuevo o mejorar lo ya existente, añadiendo nuevas características o mejorando los gráficos.

Existen multitud de tipos de juegos. Dado la gran diversidad de videojuegos resulta difícil crear una clasificación que no excluya ningún videojuego, sin embargo la inmensa mayoría debería estar incluida la siguiente clasificación o compartir suficientes rasgos como pertenecer a alguna de las siguientes.

A. Juegos Arcade:

Normalmente tienen un ritmo rápido y exigen una rápida respuesta por parte del usuario, no suelen tener componente estratégico.

- **Plataformas:** El máximo exponente de este género es el videojuego Super Mario Bros, en el que el personaje va pasando por las pantallas de un lado a otro de forma rápida eliminando enemigos.
- **Laberintos:** Este tipo de videojuegos se presenta como un laberinto en el que el personaje tiene que recorrer esquivando peligro y cogiendo recompensas, el juego más conocido de este género es el Pacman.
- **Deportivos:** Todo tipo de videojuego basado en un deporte, Futbol, baloncesto…
- **Shooter:** Todo aquel videojuego en que la mayoría de la acción se centra en disparar enemigos de una forma u otra.

B. Juegos de simulación:

En estos videojuegos los usuarios pueden tomar el control de diversos aparatos a menudo equipados con las sofisticadas tecnologías.

- **Simuladores instrumentales:** En este tipo de videojuegos se cuenta con un adaptador especial que hace creer al usuario que está tocando un instrumento musical.
- **Simuladores deportivos:** un claro ejemplo ser un videojuego de carreras de coches.
- **"Simuladores de Dios":** En estos juegos el usuario asume el papel de un ente sobrenatural que tiene una influencia sobre situación de los personajes

C. Juegos de estrategia:

En estos videojuegos la acción es más pausada, los usuarios deben encontrar la forma de ganar la partida siguiendo una estratégica, pudiendo dirigir un ejército, un negocio o escapar de una situación realizando diversas acciones.

- **Aventuras gráficas**: Es jugador vive una aventura, tendrá realizar diversas acciones para conseguir sus objetivos.
- **Juegos de Rol:** Es parecido a las aventuras gráficas, pero en este tipo de videojuegos suele implementar un sistema de niveles por los que los jugadores deberán ir pasando además de ir recogiendo diversos objetos que le ayudaran en sus tareas.

D. Juegos de mesa:

Es esta categoría estarían incluido todos los juegos de cartas, juegos de grupo como el parchís o el trivial, etc.

E. Juegos educativos:

De todos los tipos de juegos este es el menos desarrollado. Estos juegos persiguen una finalidad didáctica, a su vez pueden pertenecer a alguna de las anteriores categorías.

Este tipo de videojuegos suele estar dirigido a un público joven. Pueden ser por ejemplo un videojuego de pintar y colorear, un videojuego que enseñe matemáticas o historia o busque algún otro conocimiento.

2.2.3 Los videojuegos en red

Los videojuegos en red tienen cada día más importancia, cada día miles de personas juegan a videojuegos online como World of Warcraft, en estos juegos los usuarios compiten entre sí o colaboran para poder hacer frente a objetivos que no podrían lograr solos. En el fondo, se trata de una innovación en el ámbito de los videojuegos de "toda la vida", en los que el objetivo era enfrentarse y derrotar a la máquina en solitario.
Otra diferencia con los videojuegos tradicionales es el encuentro del usuario con otros usuarios del mundo entero pertenecientes cualquier nivel cultural o socioeconómico, lo que puede servir a los usuarios para a ampliar su visión del mundo.

2.2.4. Videojuegos, educación y desarrollo intelectual

Diversos estudios como el desarrollado por Asociación Española de Distribuidores y Editores de Software de Entretenimiento (ADESE) [6], indican que los videojuegos pueden ser una herramienta muy útil en la educación de los niños, aumentan su motivación, participación y concentración en las actividades.
De este informe se deduce que muchos videojuegos favorecen el desarrollo de determinadas destrezas que ayudan al desarrollo intelectual de la persona.
8 de cada 10 profesores considera que los videojuegos pueden ser una herramienta eficaz, especialmente con alumnos de 9 a 12 años y en asignaturas como conocimiento del medio, matemáticas e inglés.

Desde el punto de vista educativo podemos destacar los siguientes puntos.

- Carácter lúdico
- Dificultad creciente pero adaptada a los usuarios,
- Estimulación en múltiples aéreas: Visual, Auditiva, etc.
- Premiación instantánea, en forma de bonnus, logros etc.

La científica Susan Greenfield estudio el comportamiento de niños de entre 12 a 16 años y extrajo las siguientes conclusiones.

- Aumentaban las estrategias de lectura visual de imágenes y de lectura del espacio tridimensional.
- Ayudaban a trabajar el aprendizaje por observación y la verificación de hipótesis.
- Facilitaban la comprensión de simulaciones científicas.
- Incrementaban las estrategias para recibir y procesar información recibida

2.3 APLICACIONES PARA MÓVILES

Hoy en día las aplicaciones para móviles juegan un papel muy importante en nuestra vida cotidiana, existen aplicaciones para casi cualquier cosa. Un aspecto importante a tener en cuenta es que las aplicaciones deben mantener un alineamiento funcional lo más simple y claro posible, ayudando al usuario en algún aspecto de su vida cotidiana. Cuanto más específico sea el programa mejor se podrán centrar los esfuerzos de producción consiguiendo un producto de calidad.

El número de Smartphones sigue aumentando y disponer de una aplicación móvil se convertirá en algo imprescindible para cualquier empresa, como lo fue en su día disponer de una página web.

2.3.1 Internet móvil.

Gracias a los Smartphones los usuarios que tengan contratada tarifa de datos o dispongan de una red WIFI a su disposición pueden estar conectados a internet de alta velocidad, lo que brinda un mundo de posibilidades a los desarrolladores de aplicaciones móviles, tales como búsqueda de información útil, descarga de archivos adicionales, conexión con bases de datos etc.

2.3.2 Redes Sociales.

La aplicaciones para móviles pueden contar con una conexión a las redes sociales, de esta manera las aplicaciones pueden poner información en las redes sociales en las que el usuario este registrado para por ejemplo, anunciar un nuevo record en la aplicación, o invitar a otras personas a que prueben dicha aplicación.

2.3.3 Ventajas de las aplicaciones móviles.

Las aplicaciones para móviles cuentas con distintas ventajas asociadas a la propia naturaleza de la plataforma que se detallan a continuación:

- **Movilidad:** Las aplicaciones para móviles pueden ser ejecutadas en cualquier momento y en cualquier situación en la que se disponga de un Smartphone.

- **Flexibilidad:** Las tabletas son cada día más importantes en el mercado, estas tabletas ejecutan el mismo sistema operativo que sus hermanos pequeños los móviles, al crear un programa para una plataforma como Android, estaremos diseñando una aplicación que funcionara tanto en móviles como en tabletas, aunque debido a las diferencias de tamaño de estas, es conveniente rediseñar la interfaz gráfica para que tenga una correcta visualización en ambos dispositivos.

2.3.4 Tipos de aplicaciones móviles

Existen distintos tipos de aplicaciones desde el punto de vista de la programación: aplicaciones nativas (apps), páginas web y soluciones híbridas. Las aplicaciones nativas están escritas en el lenguaje que ofrece la plataforma para la creación de aplicaciones, típicamente un kit para desarrollo de software (SDK), mientras que las basadas en tecnologías web utilizan lenguajes del tipo HTML, XTHML, CSS o XML, entre otros. Adicionalmente, existen aplicaciones que combinan la parte de aplicación nativa con una parte basada en web, que podemos denominar soluciones hibridas. Todo apunta a que este tipo de aplicaciones móviles tienden a ser las más utilizadas.

I) Aplicaciones Nativas:

Son aplicaciones específicas de una plataforma, las aplicaciones desarrolladas de esta forma, solo podrán ejecutarse en un determinado sistema operativo. Por ejemplo las aplicaciones desarrolladas para iPhone solo podrán ser ejecutadas en sistema operativo iOS y viceversa en Android.

Este tipo de aplicaciones se crean con distintos tipos de lenguajes. Si la aplicación fue creada para iPhone estará programada en los lenguajes: Objective C, C, o C++, si está desarrollada para Android estará programada en Java. De modo que si el desarrollador quiere que su aplicación esté disponible en las dos plataformas, deberá hacer frente a dos desarrollos distintos ya que sus componentes están diseñados de forma específica para el sistema operativo. Pudiendo utilizar el hardware específico del sistema en donde va a ejecutarse como los sensores y elementos del teléfono: cámara, GPS, acelerómetro, agenda, etc. A continuación se muestra una figura con los sistemas operativos mayoritarios para dispositivos móviles.

Lenguajes	Obj-C, C, C++	Java (Algo de C, C++)	Java	C#, VB, NET, etc
Herramientas	Xcode	Android SDK	BB Java Eclipse Plug-in	Visual Studio, Windows Phone Dev Tools
Ejecutables	.app	.apk	.cod	.xap
Stores	Apple iTunes	Google Play	BlackBerry App World	Windows Phone Market

Ilustración 2. Comparativa Sistemas Operativos.

Ventajas de aplicaciones nativas

Tienen un rendimiento superior al estar codificadas en lenguaje nativo de la plataforma y como se ha dicho antes, las aplicaciones nativas tienen acceso total a las utilidades del sistema operativo del dispositivo: dispositivos de almacenamiento, cámara, GPS, acelerómetro, etc. Esto hace que la experiencia de usuario sea la más completa.

Además del total acceso a los elementos del teléfono las aplicaciones nativas no requieren de conexión web para ser ejecutadas (aunque esto tiende a cambiar). Por último es importante destacar que las aplicaciones nativas tendrán mucha más visibilidad ya que se distribuyen a través de la tienda de aplicaciones de los fabricantes.

Desventajas de las aplicaciones nativas:

Existen sin embargo algunas desventajas a la hora de elegir esta opción. Al estar desarrolladas para un dispositivo específico quedan fuera de su potencial mercado numerosas aplicaciones que tengan como requisito ser multiplataforma. La empresa Google en caso de Android y Apple en caso de iOS tienen que aprobar la aplicación para tenerla accesible al gran público. Las empresas necesitas desarrolladores con conocimientos específicos de la plataforma.

II) Aplicaciones Web Móviles.

Las aplicaciones web móviles, a diferencia de las aplicaciones nativas, se ejecutan dentro del navegador del teléfono. Por ejemplo, en la plataforma iOS, se ejecutan en el navegador Safari. Estas aplicaciones están desarrolladas con HTML, CSS y Javascript.

Tienen la ventaja de que al ejecutarse en el navegador, podrán ser ejecutadas en cualquier plataforma que posea un navegador: PC, iOS, BlackBerry, Android, etc. Sin embargo al tener que ser ejecutadas en el navegador, no podrán tener acceso a todos los recursos del dispositivo y tendrán un rendimiento considerablemente inferior.

Ventajas de aplicaciones Web Móviles

Al contrario que las aplicaciones nativas, las aplicaciones web se pueden ejecutar en múltiples dispositivos evitando así las complejidades de tener que crear varias aplicaciones. El proceso de desarrollo es más sencillo ya que emplean tecnologías ya conocidas como HTML, CSS y Javascript. Estas aplicaciones se pueden encontrar con los tradicionales buscadores y no necesitan de la aprobación de ningún fabricante para ser publicadas.

***Desventajas* de aplicaciones Web Móviles.**

El acceso a los elementos del teléfono son limitados, no podrán contar con aceleración de gráficos por hardware, ni tendrán acceso a los dispositivos de almacenamiento ni a los sensores, etc. Además, estas aplicaciones no se pueden vender en los MarketPlace del propietario de la plataforma, su rendimiento es inferior y con peor calidad gráfica.

III) Aplicaciones Híbridas.

Las aplicaciones híbridas aúnan lo mejor de los dos anteriores modelos. Este tipo de aplicaciones permite el uso de tecnologías multiplataforma como HTML, Javascript y CSS pero permiten acceder a buena parte de los dispositivos y sensores del teléfono. Una gran parte de la infraestructura es tipo web y la comunicación con los elementos del teléfono se hace mediante comunicadores tales como Phonegap [7]. Un buen ejemplo de aplicaciones híbridas es Facebook. Se descarga de la Store y cuenta con todas las características de una aplicación nativa pero requiere ser actualizada ocasionalmente. Gran parte de su código, la parte escrita con tecnologías multiplataforma puede ser reutilizada en sus diferentes versiones para iOS y Android.

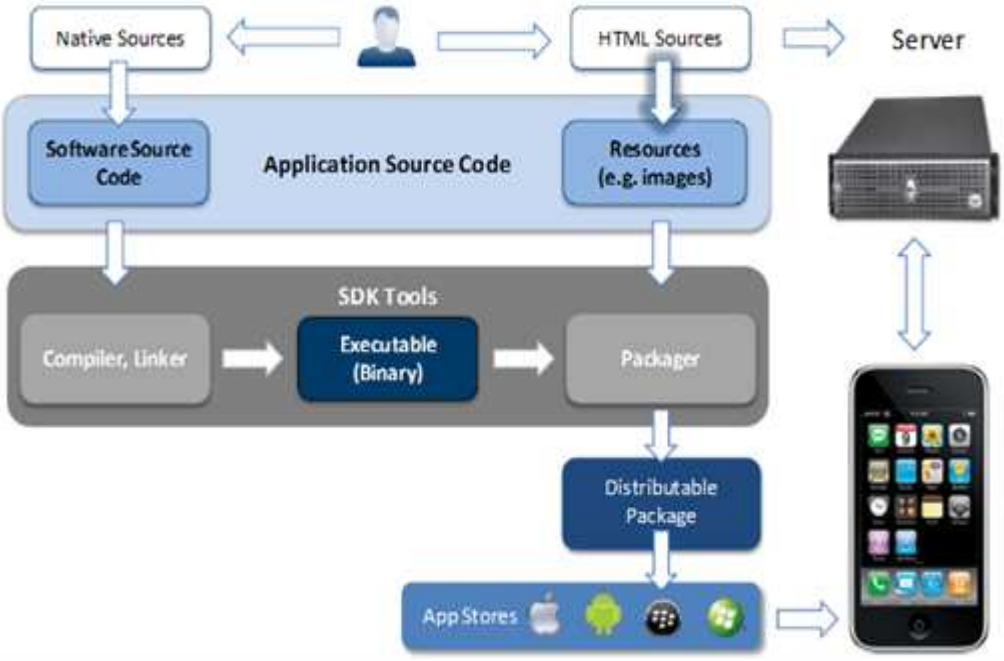


Ilustración 3. Estructura aplicaciones Web móviles.

Ventajas de aplicaciones Hibridas

Aúnan lo mejor de las dos opciones minimizando sus desventajas, pueden reutilizar parte del código, permitiendo el desarrollo de aplicaciones para nuevas plataformas rápidamente y permiten utilizar todos los recursos de los sistemas en donde se ejecutan.

Desventajas de aplicaciones Hibridas

Siguen necesitando la aprobación de la empresa propietaria del Market Place. Se sigue necesitando personal experto en las plataformas aunque en este caso el código escrito con código nativo se reduce drásticamente.

2.4 SISTEMAS OPERATIVOS PARA DISPOSITIVOS MÓVILES

Nuestra forma de vida diaria ha cambiado en los últimos años por el uso de Gadgets, pequeños aparatos que pueden facilitarnos la vida, y que acumulan una gran cantidad de funciones en pequeños dispositivos. Hoy en día una persona que tenga un Smartphone, puede tener en su mano una cámara de fotos y de video, un GPS, un teléfono, un reproductor de video, un reproductor de mp3, un ordenador, una linterna, etc. Todo ello en un pequeño dispositivo que cabe en la palma de una mano. Esta gran versatilidad ha logrado que casi todo el que puede permitírselo lleva siempre uno encima. Con la gran cantidad de dispositivos vendidos, las principales empresas de hardware y software del mundo están volcadas en desarrollar nuevos dispositivos que tengan mayores prestaciones siendo los sistemas iOS y Android lo de mayor presencia en el mercado.

A continuación se presentaran dos figuras en las que podremos ver con perspectiva la situación de Android en el segundo cuatrimestre de 2012 respecto a porcentaje y ventas totales. Los datos se han extraído de un estudio realizado por la empresa Gartner [8].

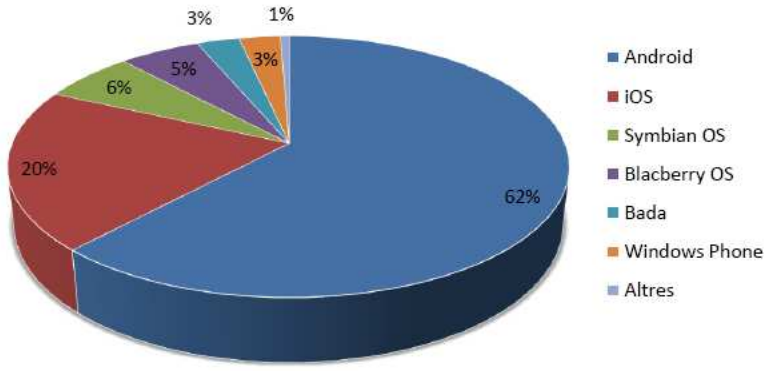


Ilustración 4: Ventas en % de sistemas operativos 2Q2012.

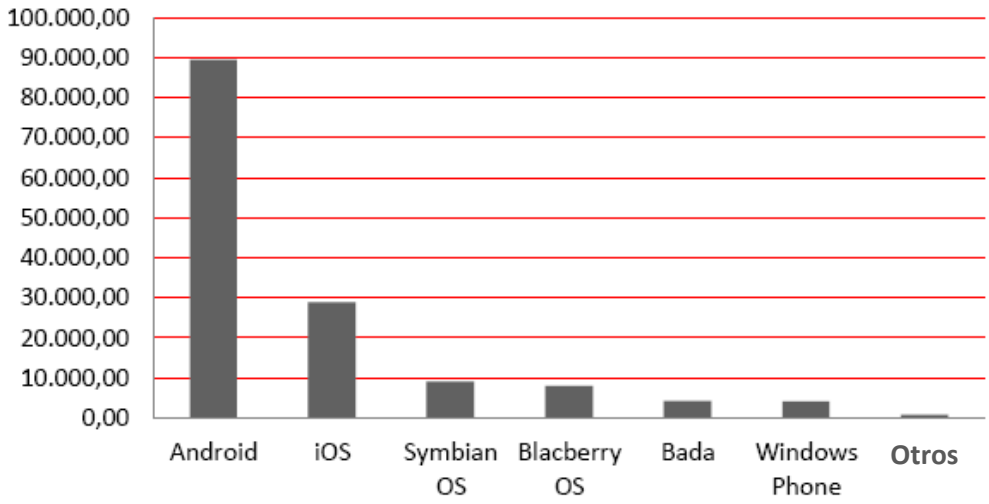


Ilustración 5. Ventas de sistemas operativos 2Q2012 (Miles de unidades).

2.4.1 iPhone OS (iOS)

Esta plataforma fue descartada en inicio al funcionar exclusivamente en terminales de gama alta incompatibles con la expectativa de uso de dispositivos de coste medio o bajo para cooperación al desarrollo. Aun así, se realiza un resumen resaltando sus características más importantes con vistas a una posible adaptación en otros proyectos futuros.

La presentación del primer iPhone supuso una auténtica revolución y supondría un cambio en el diseño de los móviles que estarían por venir. Aunque tenía grandes carencias, tenía grandes innovaciones: sus aplicaciones de correo, de navegación, reproducción de vídeos de YouTube y la revolucionaria interfaz multitáctil. Nada más salir al mercado se convirtió en todo un éxito, llegando a ser portada de la revista TIMES con el título de invento del año.

Apple pronto sacaría la App Store, su tienda digital donde se podía comprar las aplicaciones. En menos de un año desde su lanzamiento, App Store superó los 25 mil millones de descargas de aplicaciones

En 2008 salió el primer SDK para poder crear aplicaciones informáticas usando Xcode. La forma de programación en iOS es mediante C, C++ o Objective-C, iOS no permite Adobe Flash ni Java.

Características más destacadas:

- **Asistente:** Siri es un asistente creado por Apple en el que los usuarios pueden hablarle y realizarle preguntas, como por ejemplo preguntar por el tiempo o el resultado de algún partido.
- **FaceTime:** Utilidad para realizar videoconferencia, sobre 3G o WIFI.
- **Reproductor de música:** el iPhone dispone de un magnifico reproductor de música y video.3
- **Otras:** Interfaz Gráfica,Google Maps, Photo Booth, Notes, etc.

2.4.2 Android

Es un sistema operativo de código abierto diseñado para dispositivos móviles con pantalla táctil basado en un kernel de Linux.

Fue creado por Android Inc. y respaldado inicialmente por Google quien finalmente lo compró en 2005. Es producto principal de la Open Handset Alliance, un grupo de fabricantes y desarrolladores de hardware y software. Cuya finalidad es satisfacer la necesidad de los operadores móviles y fabricantes de dispositivos, además de fomentar el desarrollo de aplicaciones, cualidad que ningún otro sistema operativo incluye en sus conceptos. El primer dispositivo con Android 1.0 salió a la venta en 2008.

Pese a tener poco tiempo de vida comparado a otras sistemas operativos, su crecimiento ha sido vertiginoso hasta convertirse en 2011 en plataforma dominante por unidades vendidas tan solo tres años después de su lanzamiento.

Está basado en Linux y hereda muchas de sus características como seguridad, gestión de memoria, gestión de procesos, etc. Android es un sistema operativo de código abierto, lo que significa que cualquier fabricante puede utilizar y modificar este sistema operativo. Este hecho junto con su calidad y versatilidad son los principales factores del éxito de este sistema operativo. Pero la posibilidad de modificación no solo es accesible a los fabricantes, cualquier persona puede tener acceso al código fuente de Android y modificarlo, ya hay desarrollos independientes de Android hechos por la comunidad como Cyanogenmod, algo no disponible en sistemas privativos como el del iPhone. Los principales rivales de Android son el sistema operativo iPhone OS de la empresa Apple y el Blackberry OS de la empresa RIM, que han liderado el mercado de los Smartphone durante la última década

Sus características principales son:

- **Diseño de dispositivo: D**ispone de una biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las **especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.**
- **Almacenamiento:** SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.
- **Conectividad:** soporta las siguientes tecnologías de conectividad: GSM/EDGE,IDEN, CDMA,EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+ y WiMAX.
- **Soporte de Java:** dispone de una máquina virtual Dalvik que está diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
- **Google Play:** catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
- **Multitarea:** las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj, a diferencia de otros sistemas

A. Evolución de Android

Desde su creación, Android no ha parado de innovar y adquirir nuevas funcionalidades, a continuación se muestra la cronología y una descripción de las nuevas funciones de sus versiones.

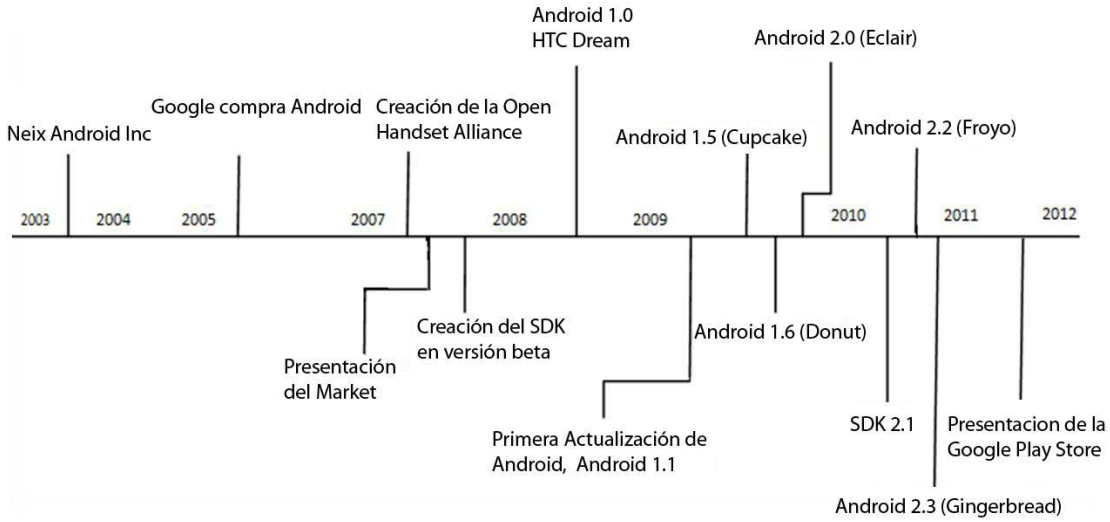


Ilustración 6: Cronograma evolución de Android 1.

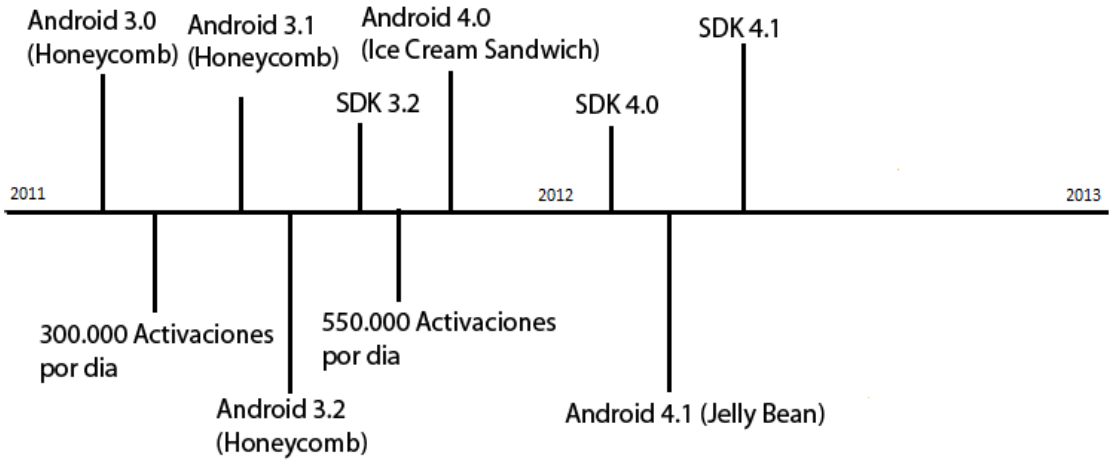


Ilustración 7: Cronograma evolución de Android 2.

1.0 Apple Pie

Lanzado el 23 septiembre de 2008 contaba con las siguientes características principales:

- Android Market Programa con un mercado para la descarga y actualización de aplicaciones.
- Navegador Web para visualizar, páginas webs en full HTML y XHTML
- Soporte Cámara – sin embargo esta versión carece de la opción de cambiar la resolución de la cámara
- Google Maps con Latitude y Street View
- Reproductor de medios, habilitada administración, importación, y reproducción de archivos multimedia
- Soporte para Wi-Fi y Bluetooth.

1.1 Banana Bread

- Detalles y reseñas disponibles cuando un usuario busca por negocios en los mapas.
- Pantalla en-llamado más larga por defecto cuando están en uso los parlantes del teléfono, además la habilidad de mostrar/esconder el discado.
- Posibilidad de guardar los archivos adjuntos en los mensajes.
- Añadido soporte para marquesina en diseños de sistemas.

1.5 Cupcake

- Soporte para teclados virtuales de terceros con predicción de texto y diccionario de usuarios para palabras personalizadas.
- Soporte para Widgets - vistas de miniaturas de las aplicaciones que pueden ser insertadas en otras aplicaciones
- Grabación y reproducción en formatos MPEG-4 y 3GP.
- Características de Copiar y pegar agregadas al navegador web.
- Agregada opción de auto-rotación.
- Habilidad de subir videos a Youtube.

1.6 Donut

- Mejora en la búsqueda por entrada de texto y voz para incluir historial de marcadores, contactos, y la web.
- Habilidad de los desarrolladores de incluir su contenido en los resultados de búsqueda.
- Motor multi-lenguaje de Síntesis de habla para permitir a cualquier aplicación de Android "hablar" una cadena de texto.
- Búsqueda facilitada y habilidad para ver capturas de las aplicaciones en el Android Market(Google Play).
- Galería, cámara y filmadora con mejor integración, con rápido acceso a la cámara.
- Actualización soporte a tecnología para CDMA/EVDO, 802.1x, VPNs, y un motor text-to-speech.
- Soporte para resoluciones de pantalla WVGA.

2.2.x Froyo

- Optimizaciones en velocidad, memoria y rendimiento
- Mejoras adicionales de rendimiento de aplicación, implementadas mediante compilación Just-in-time (JIT)
- Integración del motor de JavaScript V8 de Chrome en el navegador.
- Soporte para el servicio Android Cloud toDeviceMessaging (C2DM), habilitando notificaciones push
- Actualizada la aplicación Market con características de grupo y actualizaciones automáticas
- Discado por voz e intercambio de contactos por Bluetooth
- Soporte para docks Bluetooh-habilitado para autos y de

3.x Honeycomb

- Soporte optimizado para tablets, con una nueva y "virtual" interfaz de usuario holográfica.
- Multitarea simplificada – tocando Aplicaciones recientes en la barra del sistema permite a los usuarios ver instantáneas de las tareas en curso y saltar rápidamente de una aplicación a otra.
- Teclado rediseñado, permitiendo una escritura rápida, eficiente y acertada en pantallas de gran tamaño.
- Nueva interfaz de contactos de dos paneles y desplazamiento rápido para permitir a los usuarios organizar y reconocer contactos fácilmente.
- Nueva interfaz de correo de dos paneles para hacer la visualización y organización de mensajes más eficiente, permitiendo seleccionar uno o más mensajes.
- Soporte para videochat usando Google Talk.
- Aceleración de hardware.
- Soporte para microprocesadores multi-núcleo.
- Habilidad para encriptar todos los datos del usuario.
- Filesystem in Userspace (FUSE; kernel module).
- Conectividad para accesorios USB.
- Soporte para joysticks y gamepads.

4.0.x Ice Cream Sandwich

- Facilidad para crear carpetas, con estilo de arrastrar y soltar.
- Captura de pantalla integrada (manteniendo presionado los botones de bloqueo y de bajar volumen).
- Corrector ortográfico del teclado mejorado.
- Funcionalidad copiar-pegar mejorada.
- Mejor integración de voz y dictado de texto en tiempo real continuo.
- Desbloqueo facial, característica que permite a los usuarios desbloquear los equipos usando software de reconocimiento facial.
- Nuevo navegador web con pestañas bajo la marca de Google Chrome, permitiendo hasta 15 pestañas.
- Sincronización automática del navegador con los marcadores de Chrome.
- Capacidad para cerrar aplicaciones que están usando datos en segundo plano.
- Aplicación de la cámara mejorada sin retardo en el obturador, ajustes para el time lapse, modo panorámico y la posibilidad de hacer zoom durante la grabación.
- Android Beam, una característica de Nearfieldcommunication que permite el rápido intercambio de corto alcance de enlaces web favoritos de un navegador de internet, información de contactos, direcciones, videos de YouTube y otros datos
- Aceleración por hardware de la interfaz de usuario
- Wi-Fi Direct
- Grabación de video a 1080p para dispositivos con Android de serie.
- Android VPN Framework (AVF)
- Numerosas optimizaciones y corrección de errores

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	3.7%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	38.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1.x	Jelly Bean	16	26.1%
4.2.x		17	2.3%

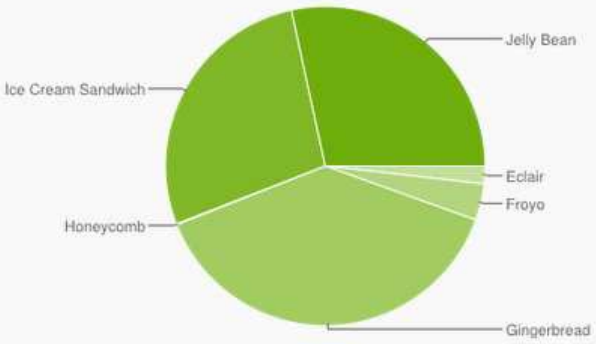


Ilustración 8: Porcentaje de versiones de Android en dispositivos activos, Abril 2013.

B. Google Play y las Apps

Android cuenta con una gran comunidad de desarrolladores. Para facilitar la tarea de distribuir esas apps, Google creo la tienda Android Market después llamada Google Play, en donde los usuarios pueden buscar, obtener información y descargar aplicaciones publicadas por otros desarrolladores, según un comunicado de Google en octubre de 2012 Google Play cuenta con más de 700.000 apps, El número de aplicaciones disponibles en una plataforma es uno de los factores a tener en cuenta para determinar su importancia. Tradicionalmente la App Store de Apple, el principal competidor, ha aventajado en este terreno a sus rivales. Una tendencia que ha ido cambiando poco a poco a favor de Android, pese al crecimiento estable del SO de Apple que cuenta con unos 750.000 apps para descargar.Google Play no solo se limita a distribuir appsademás cuenta con GooglePlay Music, Google Play Books, Google Play Magazines y Películas y Series de televisión en Google Play Movies, en varios países.

Crecimiento de Google Play

Google anunció Android Market el 28 de agosto de 2008 y lo puso a disposición de los usuarios el 22 de octubre de 2008 y se introdujo soporte para las aplicaciones de pago del 13 de febrero de 2009 .El 17 de marzo de 2009, alrededor de 2.300 aplicaciones estaban disponibles en Android Market, de acuerdo con el director técnico de T-Mobile Cole Brodman.4 El 10 de mayo de 2011, durante Google I/O, Google anunció que en Android Market figuran 200.000 aplicaciones y habían sido instaladas 4.500.000.000 aplicaciones, a continuación de muestra la evolución del número de aplicaciones disponibles.

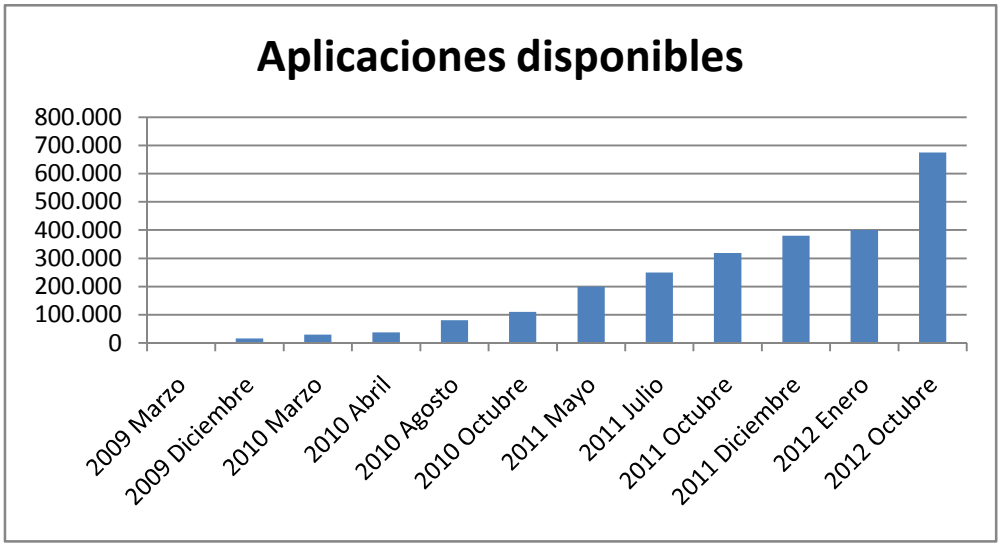


Ilustración 9: Cronograma de aplicaciones disponibles en Google Play.

En la Figura 11 se detalla la evolución de las descargas de Apps Android, entre los años 2008 y 2012 en millardos (Téngase en cuenta que un Billón Americano es equivalente a mil millones españoles, es decir a un millardo). Como puede observarse, la evolución en el número de descargas es exponencial habiendo alcanzado los 25 mil millones a finales de 2012.

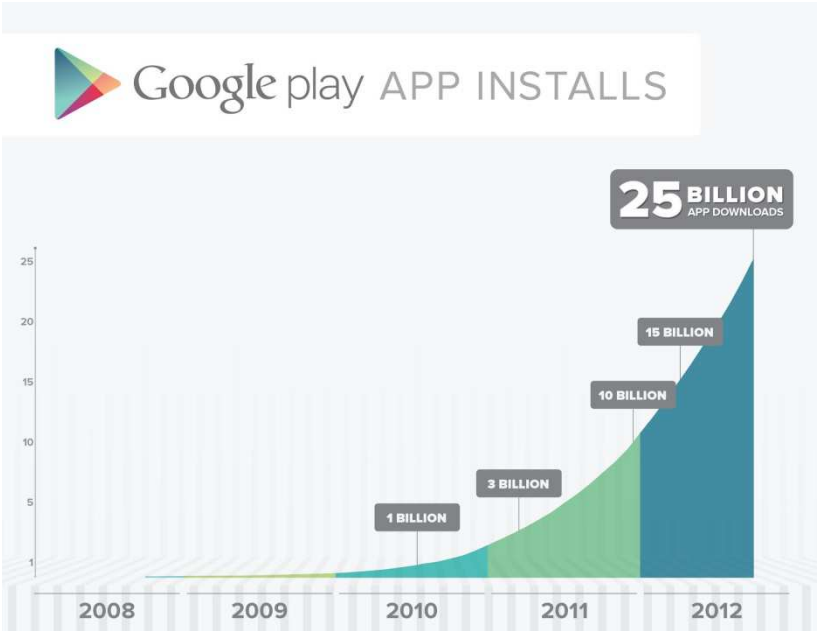


Ilustración 10. Evolución de las descargas de Apps Android [23]

C. *Java y Android*

El lenguaje Java fue elegido por Google para programar aplicaciones para Android. JAVA es un lenguaje de propósito general basado en clases y orientado a objetos cuya sintaxis deriva de C y C++. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que se pueden ejecutar en cualquier máquina virtual Java (JVM), La función de esta máquina virtual es ejecutar el código java para que el mismo código pueda funcionar en distintas plataformas. Fue creado por Sun Microsystems a mediados de los 90, inicialmente para generar aplicaciones que controlaran electrodomésticos como lavadoras, frigoríficos, etc. Por lo que se diseñó específicamente para tener tan pocas dependencias de implementación como fuera posible. El código escrito en java necesita una maquina virtual que haga de intermediario, hecho que hace que el sistema sea más lento que otro idéntico que ejecute código escrito en C y se pueda ejecutar de forma nativa. Existen muchas voces críticas sobre el uso de Java en Android e incluso existe una empresa que está desarrollando una versión de Android llamada Xobot OS que utiliza C# en lugar de Java.

En Android las aplicaciones se ejecutan en un framework Java sobre el núcleo de las bibliotecas en una máquina virtual Dalvik con compilación en tiempo de ejecución. Esto significa que el bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y este se ejecuta en la Máquina Virtual Dalvik.

Dalvik es una maquina virtual basada en registros a diferencia de la maquina virtual de Java basada en el uso de las pilas. La maquina Dalvik ejecuta el formato Dalvik Executable (*.dex) que es un formato optimizado para almacenamiento eficiente y ejecución mapeable en memoria. El recolector de basura de Java se ha perfeccionado con el fin de mantener en todo momento el máximo posible de memoria disponible. Dalvik está diseñada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. Esta optimizada para que múltiples instancias de ella puedan funcionar al mismo tiempo con un impacto muy bajo en el rendimiento de la memoria del dispositivo, de esta forma se protege el sistema. Por ejemplo si fallara una aplicación la otras seguirían con su ejecución normal al estar en diferentes instancias de la maquina virtual.

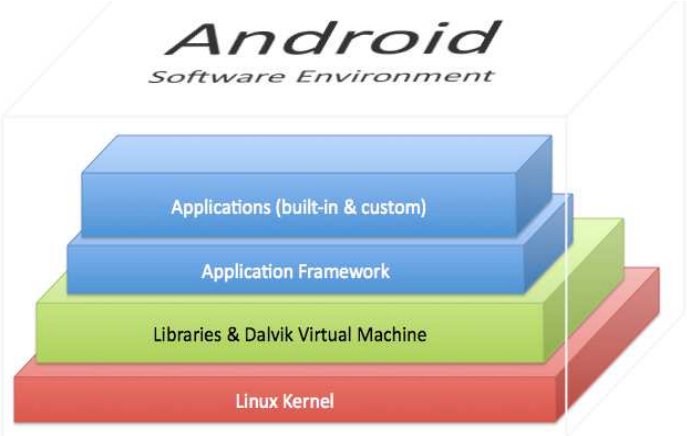


Ilustración 11: Capas de software en Android.

El objetivo de esta máquina virtual es permitir que el bytecode sea independiente de la máquina para que pueda ser utilizada por distintos dispositivos. Una desventaja de este proceso es que se necesita optimizar al máximo los recursos y enfocar el funcionamiento de los programas dado que las maquinas que utilizan este sistema son de recursos limitados.

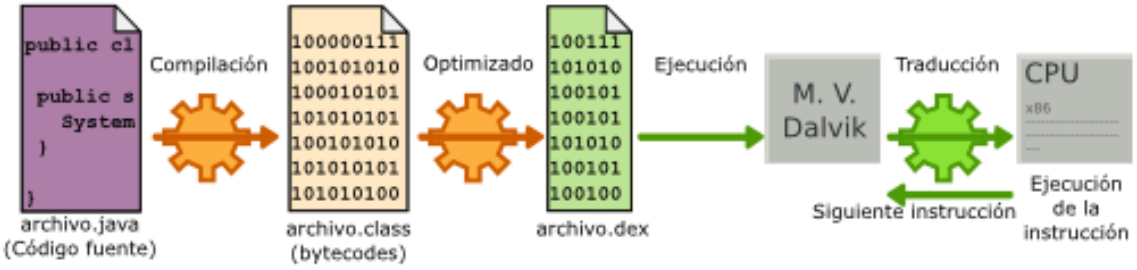


Ilustración 12: Funcionamiento de la máquina Dalvik.

Dalvik tiene unas características específicas que lo diferencian de otras máquinas virtuales estándar:

- La máquina virtual se ha diseñado para utilizar menos espacio.
- El conjunto de constantes se ha modificado para utilizar sólo los índices de 32 bits para simplificar el intérprete.
- El bytecode Java estándar ejecuta instrucciones de pila de 8 bits. Las variables locales se deben copiar hacia o desde de la pila de operandos utilizando instrucciones por separado. Dalvik utiliza su propio conjunto de instrucciones de 16 bits que trabaja directamente sobre las variables locales. La variable local se recoge habitualmente por un campo de 4 bits " registro virtual".

2.4 XML

El eXtensible Markup Language o XML es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) http://www.w3c.es que deriva del lenguaje SGML.

XML define un conjunto de normas para la codificación de documentos que es entendible por personas y ordenadores. Está diseñado para simplificar y estandarizar el intercambio de información por internet, aunque su usabilidad va más allá y puede ser utilizado en bases de datos, para intercambiar información entre distintas plataformas o codificar en el cualquier documento como por ejemplo hojas de cálculo.

Ejemplo de documento en XML:

```
- <xml>
    <title>XML test</title>
  - <text type="test">
    - <body>
      - <p>
          Though this is a very pared
          <lb />
          down XML document, it nonetheless
          <lb />
          provides an example of how an XML
          <lb />
          document displays on the web without
          <lb />
          the intercession of a stylesheet or
          <lb />
          other conversion program.
        </p>
      </body>
    </text>
  </xml>
```

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan, con todas estas tecnologías relacionadas, representa una forma distinta e innovadora de hacer las cosas por ello es una forma sencilla, fácil y segura de transmitir la información. Permitiendo a los desarrolladores dedicar su esfuerzo a otras tareas evitando tareas tediosas como la validación.

XML nació con los siguientes objetivos.

- ✓ Que fuera idéntico a HTML en la forma para poder procesar la información de la misma manera aprovechando los esfuerzos previos.
- ✓ Que sea formal y conciso para evitar ambigüedades a la hora de representar los datos
- ✓ Que fuera extensible para poder utilizarse en todos los ámbitos.
- ✓ Que fuese de fácil entendimiento por las personas
- ✓ Que fuese fácil de escribir, e implantar en los distintos sistemas.

Hay dos tipos de documentos XML: válidos y bien formados. Éste es uno de los aspectos más importantes de este lenguaje.

Los documentos bien formados son aquellos que cumplen con todas las normas que dicta el estándar y pueden ser leídos por cualquier analizador sintáctico “parser” que más adelante se explicará.

Un documento XML tienen una estructura jerárquica, que parte de un solo nodo raíz y dentro de él, las etiquetas que delimitan sus elementos se anidan unas dentro de otras, debiéndose respetar el orden jerárquico a la hora de abrir y cerrar dichas etiquetas, estas etiquetas siempre tienen que cerrarse al final.

Los documentos validos además de estar bien formados los archivos XML validos siguen una estructura y una semántica determinada por un DTD o por un Schema.

Un DTD especifica los elementos y el orden de ellos que puede contener un documento XML, básicamente es un conjunto de reglas que debe cumplir un archivo XML para ser valido respecto a un DTD especifico, Mientras que un XML Schema es un lenguaje de esquema que sirve para especificar la forma concreta que debe seguir el archivo XML de la misma forma que un DTD, pero tiene más ventajas:

- Está escrito en XML al contrario que los DTDs.
- Permite especificar los tipos de datos
- Son extensibles

Los programas que leen los documentos XML son los parsers. Los parsers son programas informáticos que se encargan de transformar el texto leído en otras estructuras generalmente arboles, permitiendo navegar por la estructura y obtener la información deseada. Hay dos tipos, los validadores que validan la información leída respecto a un DTD y no validadores que simplemente leen la información si está bien formado [9].

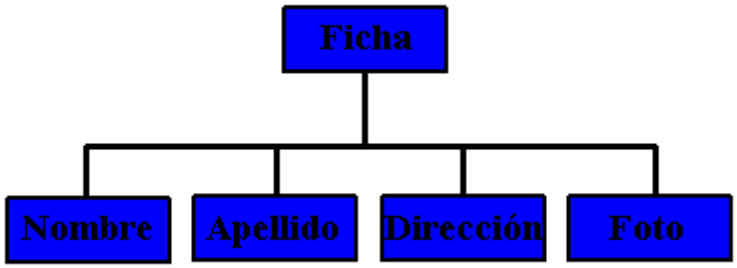


Ilustración 13: Documento en forma de árbol.

2.6 ANDENGINE

AndEngine es un framework para la creación de juegos 2D, utiliza GLES2 y también cuenta con un motor de física llamado Box2D que es el mismo utilizado en el superventas Angry Birds.

Fue creado Nicolas Gramlich en 2010 bajo licencia LGPL [10], la información que se puede encontrar de este framework es realmente escasa. De hecho este es su principal hándicap y una crítica omnipresente en todos los foros. La documentación disponible sobre este framework consta de una docena de ejemplos pequeños que no cubren ni el 10% de sus posibilidades. Como consecuencia, puede verse por los testimonios en los foros que la inmensa mayoría de usuarios abandona el uso de este framework a la pocas horas o días.

Sin embargo debido a su gran potencia en comparación con otros frameworks mucho mejor documentados, mucha gente elige este framework para desarrollas sus aplicaciones.

Una de las principales factores que llevaron a la utilización de un framework como este y no programar directamente la aplicación para Android es la falta de aceleración Hardware por parte de Android de los Sprites (un tipo de mapa de bits dibujados en la pantalla), AndEngine utiliza Open GL ES, lo que le permite utilizar la potencia de la tarjeta grafica del dispositivo, permitiendo la ejecución de esta aplicación a velocidades de refresco superiores a 40 fps en dispositivos de gama baja como el Samsung Galaxy Mini utilizado para las pruebas.

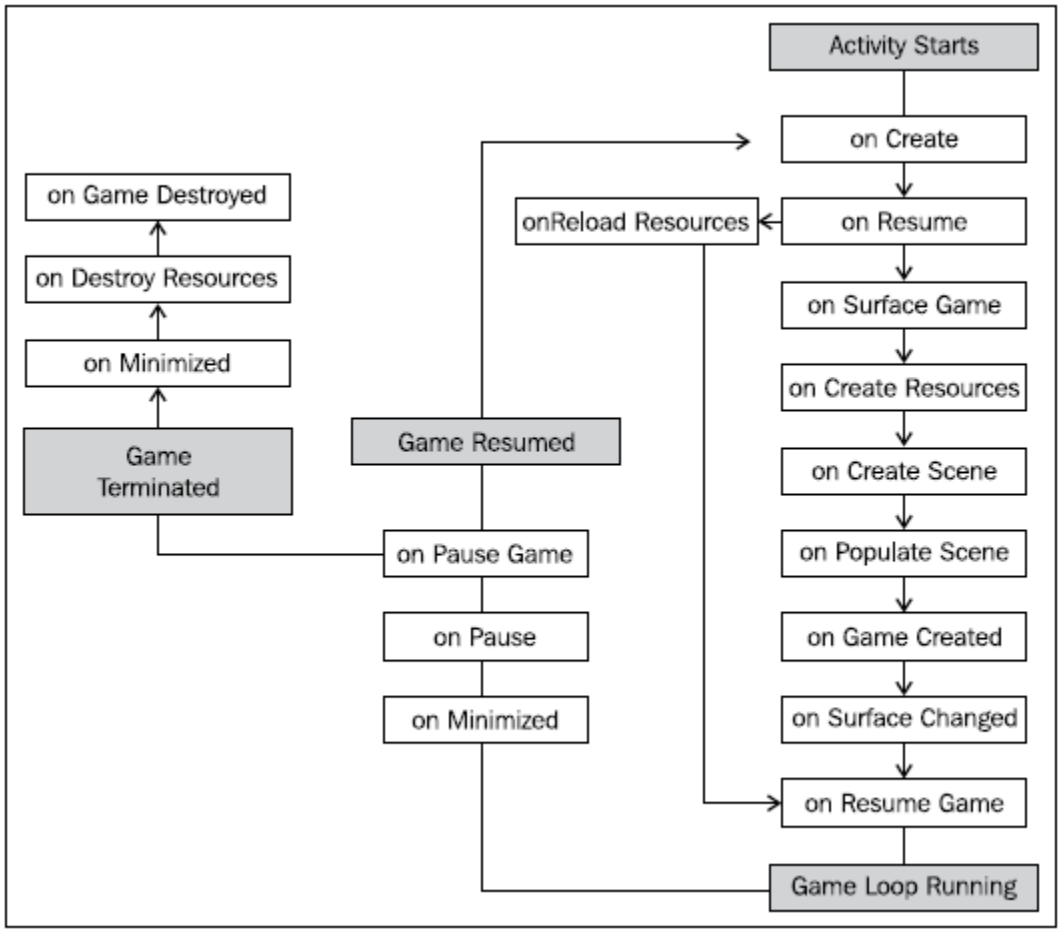


Ilustración 14: Ciclo de vida de una Aplicación realizada con AndEngine

2.7 SÍNTESIS

El espíritu de este PFC es la difusión del IDH, así como de la educación y concienciación de los usuarios de este PFC. Por ello se decidió que se deberían tomar las decisiones que llevaran este PFC al mayor público posible.

La creación de una aplicación móvil es una forma idónea para llegar a un gran número de usuarios ya que cada día son más importantes, en la actualidad los Smartphones son los teléfonos más vendidos en gran parte gracias a las aplicaciones diseñadas para estos que son cada vez más numerosas y diversas, si sumamos este hecho junto a que existen 7000 millones de abonados móviles en el mundo, según la Unión Internacional de Telecomunicaciones ITU [12]. Este hecho junto con la Existencia de las Market place para las plataformas para dispositivos móviles, hacen que sea fácil que un pequeño desarrollador pueda poner su aplicación disponible a todos los usuarios de la plataforma.

La decisión de que el tipo de aplicación debía ser un videojuego se debió a que la mejor forma posible de presentación de una aplicación que no tenga una utilidad funcional específica, como puede ser una aplicación de localización de GPS, o un nivel, o una aplicación de linterna. Era que debía ser divertida para que los usuarios se interesaran por la aplicación, y pasaran tiempo con ella, ya que se necesita un tiempo para que los usuarios tuvieran una concienciación con el IDH y lo que representan sus cifras.

Una vez decidido a que de debería realizar un videojuego para una plataforma móvil se decidió por Android ya que es la extendida a todos los niveles socioculturales. iOS es una plataforma mucho más privativa, cuyo coste supera los 600 Euros y en la que los precios de los terminales son de los más elevados. Sin embargo existe mucha diversidad de dispositivos con Android, muchos de ellos con precios inferiores a 200 Euros.

3 ANÁLISIS DEL SISTEMA

En este capítulo se presenta el sistema de juego cuyo análisis de detalla mediante la utilización de diagramas UML (Unified Modeling Language).

3.1 DESCRIPCIÓN GLOBAL DEL SISTEMA

El sistema estema está formado principalmente por dos sistemas: el Cliente, aplicación móvil construida en este PFC, y el servidor de partidas desarrollado en un PFC complementario que sirve para que los diversos usuarios de este PFC puedan jugar Online entre ellos.

3.1.1 El Juego

La aplicación se presenta como un videojuego de cartas multijugador online, cuya temática está centrada en el IDH de los países del mundo. Los jugadores juegan con cartas que representan cada país y sus datos sobre su población: IDH, ingresos, educación y salud.

Se reparten 49 cartas por jugador, una vez repartidas se tira un dado que determinara que jugador empieza la ronda, El jugador que empiece la ronda deberá elegir una carta que se mostrará a los demás jugadores, indicando que dato es el elegido para jugar, pero no verán el valor de dicho dato, Los demás jugadores podrán elegir las cartas con las que quieren jugar y tendrán que usar sus conocimientos para juzgar el valor de ese dato del país del jugador que empezó la ronda, he intentar elegir una carta que pueda superar dicho valor.

Cuando todos los jugadores han elegido sus cartas, se revelan dichos valores y el tenga un valor más alto ganara todas las cartas que estén en juego las caras y empezará la siguiente ronda. Una vez se hayan jugado todas las rondas, ganara el jugador que tenga una mejor puntuación de IDH en sus cartas ganadas.

El sistema de juego premia a los jugadores con mayores conocimientos sobre los datos humanos de los diferentes países del mundo, de ese modo los mejores jugadores serán los que tengan más conocimientos de estos datos.

3.1.2 La Aplicación Móvil

La aplicación móvil está programada en java para Android. Para ello, utiliza un Framework llamado AndEngine, que consta de una serie de clases que sirven de capa intermedia entre Android y la aplicación, Este Framework aporta Aceleración Hardware, mediante la utilización de Open GL ES 1.0, esta aceleración permite que móviles viejos o no muy potentes puedan ejecutar la aplicación con un rendimiento suficiente, debido a la utilización de Open GL se puede modificar la perspectiva de la cámara para que ofrezca imágenes en 3D como se ha hecho en esta aplicación.

Se ha desarrollado un motor de razonamiento sencillo suficiente para el desarrollo normal de una partida mono jugador. Cuenta también con comunicación vía internet con un servidor que permite que diferentes usuarios puedan jugar juntos en la misma partida.

3.1.3 El Servidor de partidas

Es un programa informático escrito en java que escucha en un puerto, cuando un usuario de la aplicación móvil selecciona que quiere jugar una partida por internet, la aplicación se conecta al servidor y es encolada a la espera de otro jugador, cuando tiene a dos jugadores el servidor inicia la partida comunicando a los dos jugadores entre sí.

Cuenta además con una base de datos de registro de usuarios que sirve tanto para la pagina web y el foro como para las partidas multijugador de la aplicación, los usuarios deben registrarse antes de poder jugar, cuando quieran jugar deben facilitar al servidor su usuario y contraseña, este dato quedara guardado en el terminal para futuras partidas.

3.1.4 La pagina Web

En la página web se pueden leer noticias sobre el IDH, dispone de un foro donde los usuarios pueden debatir temas relacionados con el IDH o de otros temas, también permite la descarga de la aplicación móvil y permite a los usuarios acceder a sus estadísticas y ver su ranking.

3.2 ARQUITECTURA DEL SISTEMA

El sistema está compuesto por tres entidades:

- **Servidor:** Un ordenador conectado a Internet que almacenara una página, este servidor integra el programa servidor para las partidas multijugador.
- **Base de datos:** Se encuentra en el servidor, almacenara a los usuarios registrados y sus datos.
- **Terminal móvil**: Un dispositivo inteligente como un Smartphone o una tablet en donde se ejecuta la aplicación, debe disponer de conexión a internet: WIFI, GPRS, 3G… para poder disfrutar de las opción de Multijugador.

La figura siguiente explica las interacciones entre las entidades del sistema cuyas comunicaciones se realizan según el protocolo OTA (Over-the-air) usado en telefonía móvil:

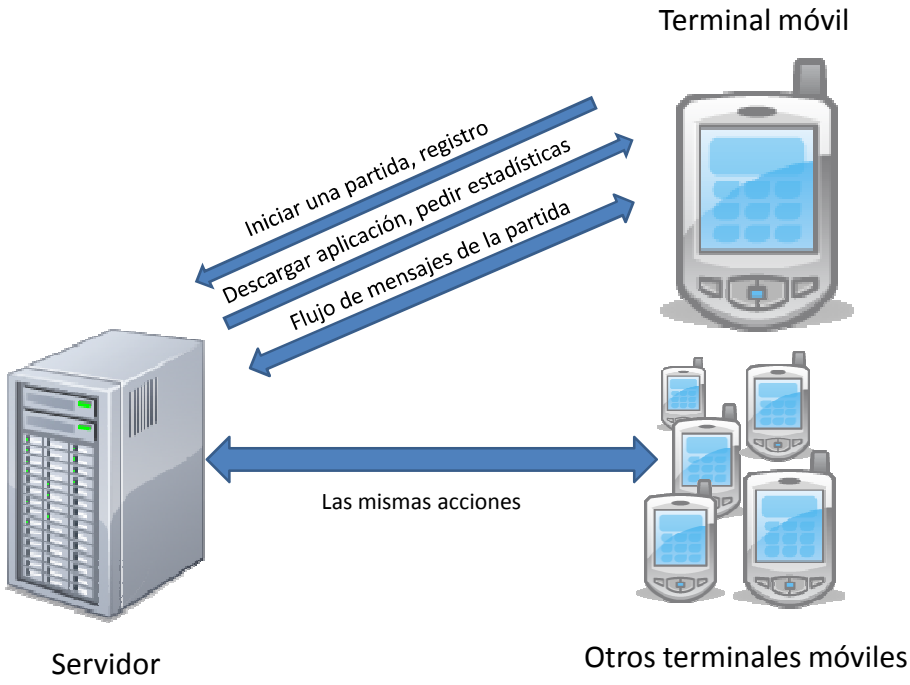


Ilustración 15: Arquitectura de la plataforma.

A continuación podemos ver un diagrama de casos de clases teórico de la aplicación móvil. En el apartado de diseño se mostrará el diagrama de clases refinado que finalmente ha sido implementado en conexión con el diagrama de la aplicación del servidor.

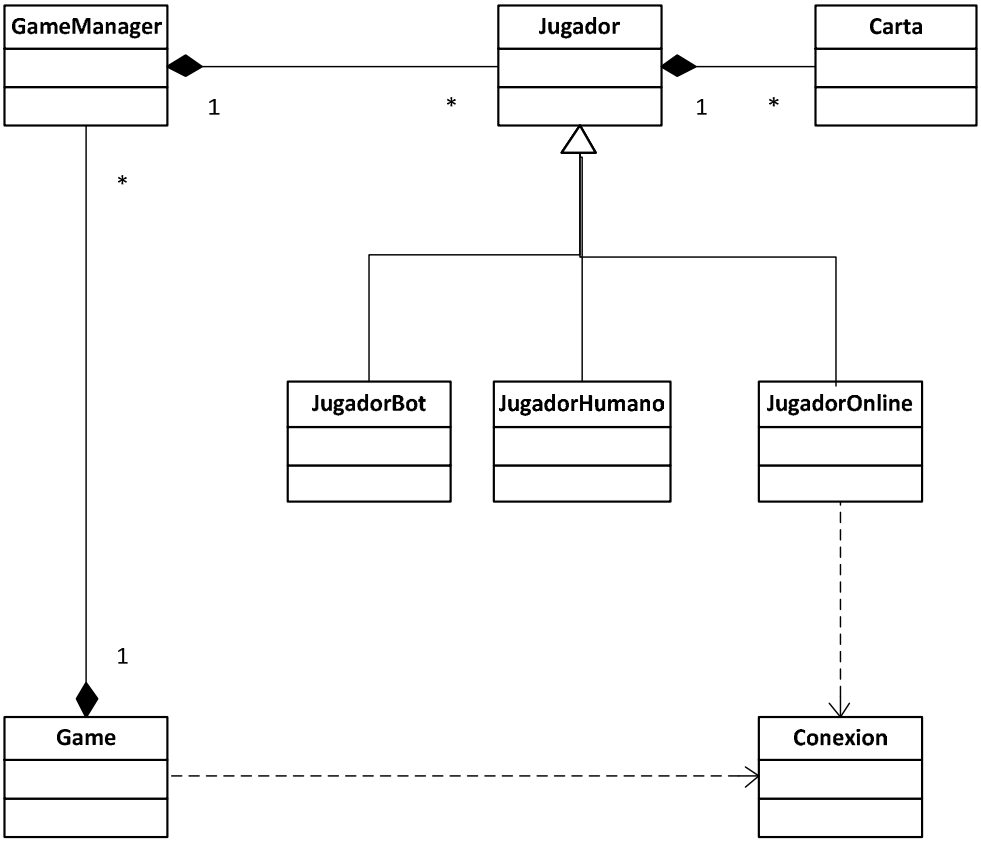


Ilustración 16: Diagrama de clases teórico.

3.3 ESPECIFICACIÓN DE REQUISITOS

En este apartado se especifican los requisitos del sistema, tanto funcionales como no funcionales.

I) Requisitos funcionales

I.1. **Fácil localización y descarga:** La aplicación tiene que ser fácil de encontrar por ello se puede descargar tanto de la página web como de Google Play.

I.2. **Permitir el registro:** La aplicación permite el registro del usuario en un formulario.

I.3. **Modificación de datos:** Los usuarios deben poder cambiar sus datos, por ejemplo de la contraseña.

I.4. **Partida mono-Jugador:** La aplicación debe permitir que el usuario pueda jugar una partida sin necesidad de conexión a internet.

I.5. **Partida online:** La aplicación debe permitir que los usuarios de cualquier parte del mundo puedan jugar en línea.

I.6. **Acceso a los datos**: La aplicación debe contar un apartado en donde los jugadores puedan consultar los datos humanitarios de todos los países.

II) Requisitos no funcionales

II.1. **Buena apariencia visual:** Para que tenga una buena aceptación debe contar con una buena apariencia estética.

II.2. **Fácil manejo:** La aplicación debe ser intuitiva y fácil de utilizar.

II.3. **Estabilidad y compatibilidad:** debe guardar compatibilidad con la mayor parte de las versiones de Android, y debe ser estable y no tener errores.

II.4. **Bajos recursos de hardware:** debe mantener unos requisitos de hardware bajos para permitir que la mayor parte de dispositivos puedan ejecutar la aplicación.

3.4 CASOS DE USO

A continuación se detallan las acciones que permite la aplicación al usuario mediante la utilización de diagramas de casos de uso. Los actores del sistema son:

Usuario: es la persona que utiliza la aplicación en su dispositivo móvil.

Servidor: Programa informático, su función es la de la registro de usuarios en su base de datos y posibilitar el modo de juego online.

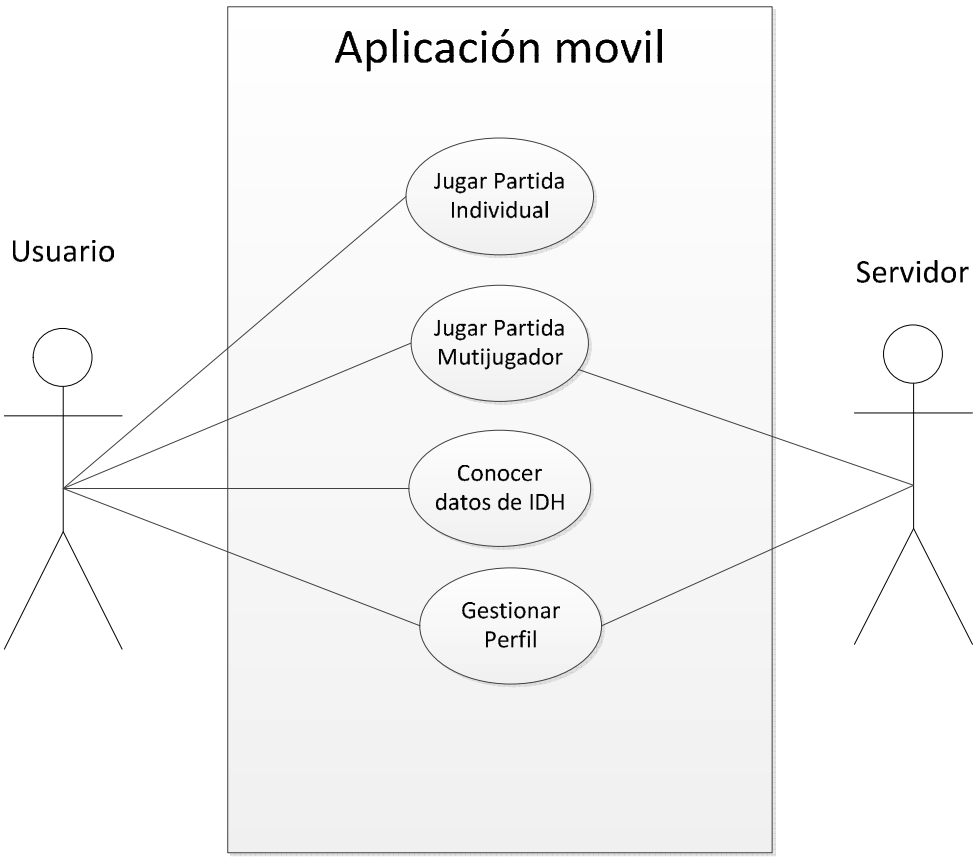


Ilustración 17: Casos de uso aplicación móvil.

Casos de Uso:

- CU1. - Jugar partida Individual: El jugador podrá jugar contra otros jugadores entre 1 y 3 adversarios gobernados por la Inteligencia artificial.
- CU2. - Jugar partida Multijugador: En este modo de juego la aplicación utiliza el servidor para buscar otros jugadores en línea para unirlos en una partida.
- CU3. - Conocer datos de IDH: El usuario puede ver todas las cartas de modo que pueda estudiarse los datos para jugar con más información.
- CU4. - Gestionar perfil: Los usuarios podrán registrarse, una vez registrados tendrán la posibilidad de modificar los datos que ingresaron, como por ejemplo el país el nombre o la contraseña.

El Caso de Uso 4, Gestionar perfil se divide en tres casos de uso más específicos.

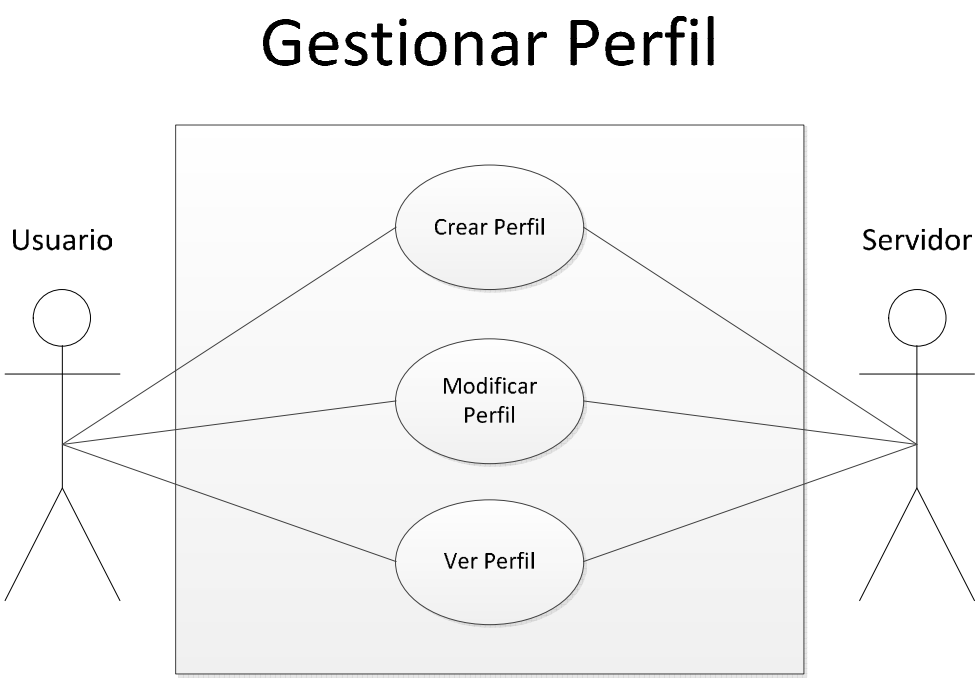


Ilustración 18: Caso de uso gestionar perfil.

CU4.1- Crear Perfil: el usuario debe introducir sus datos así como de una dirección de correo que permitirá recuperar la contraseña si se olvida la aplicación recordara siempre el último usuario y contraseña introducido para las siguientes partidas.

CU4.2-Modificar Perfil: permite cambiar los datos personales así como de la contraseña introduciendo la contraseña anterior.

CU4.3-VerPerfil: el caso de uso visualizar perfil permite al actor usuario obtener información sobre sus datos de registro (nombre, apellidos, nacionalidad, usuario) y sobre los resultados de las partidas realizadas.

3.5 DIAGRAMAS DE SECUENCIA

Estos diagramas muestran la secuencia de acciones y el intercambio de mensajes entre el servidor y los clientes.

3.5.1 Registro de usuario.

En la figura siguiente se detalla el escenario en el que la aplicación realiza el registro de un usuario, para ello el usuario deberá introducir en la aplicación datos tales como Nombre, usuario, contraseña etc. Una vez el servidor ha validado que no existía previamente dicho usuario, lo almacenara en la base de datos.

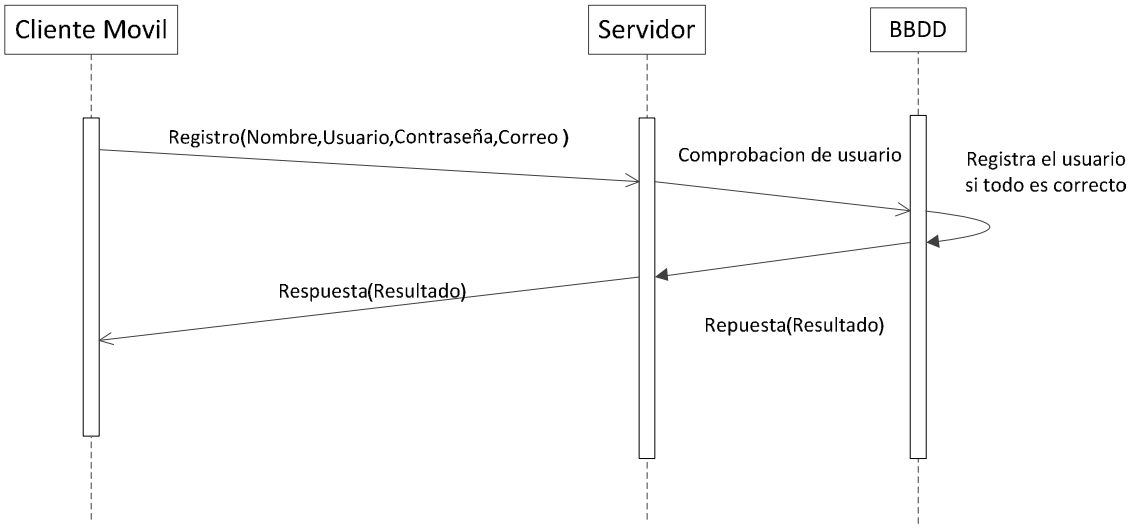


Ilustración 19: Diagrama de secuencia de registro.

3.5.2 Ver el perfil de usuario

El usuario puede consultar sus datos, para ver si están correctos o por si hubiera dos personas que utilizan la aplicación con el mismo terminal, esta función serviría para mostrar al usuario con que cuenta esta activa en la aplicación.

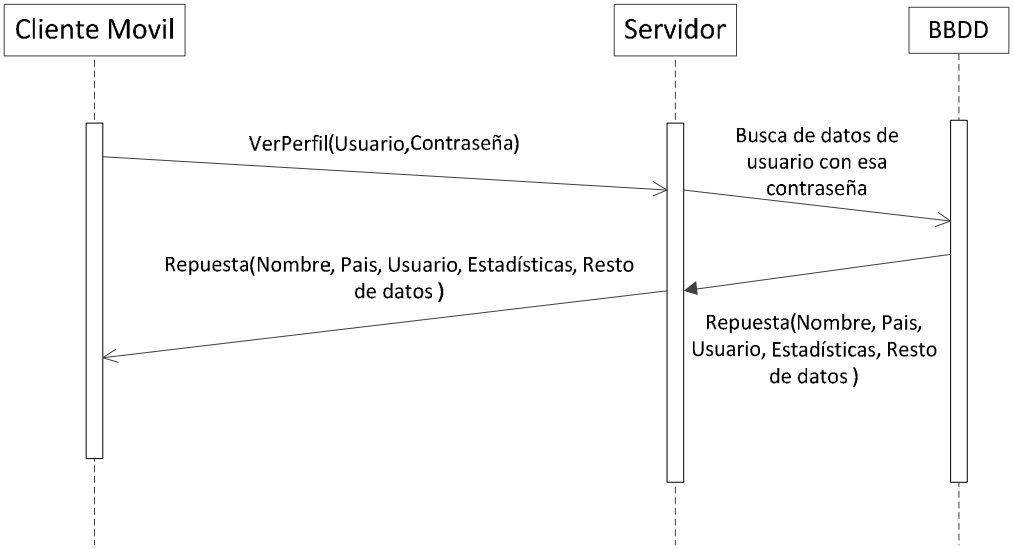


Ilustración 20: Diagrama ce secuencia de visualización de perfil.

3.5.3 Modificación perfil

En la siguiente figura se observa el intercambio de mensajes y las acciones necesarias para la modificación de los datos personales del usuario que lo ha solicitado. El usuario modifica los datos que había introducido anteriormente y la aplicación se los envía al servidor. El servidor comprueba que el usuario y la contraseña son correctos, en el caso de que el usuario modificara la contraseña, en los datos se enviara la nueva contraseña junto con la antigua para validar el proceso. Una vez se completo el proceso con éxito se le enviaran los datos al cliente a modo de verificación.

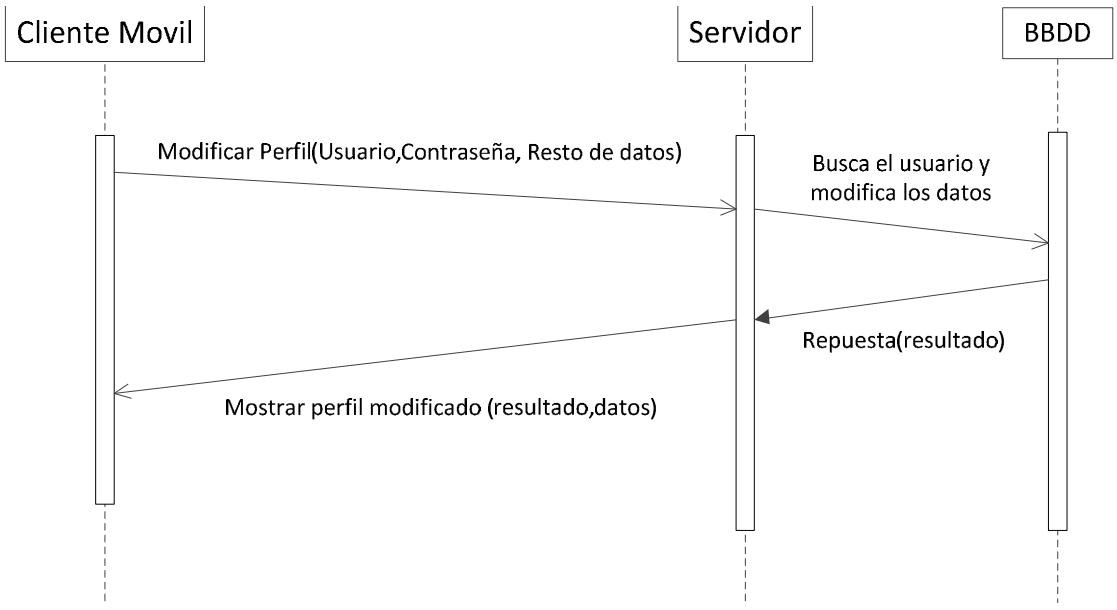


Ilustración 21: Diagrama de secuencia de modificación de perfil.

3.5.4 Login.

La aplicación inicia un proceso de Login en el servidor previa a una partida online.

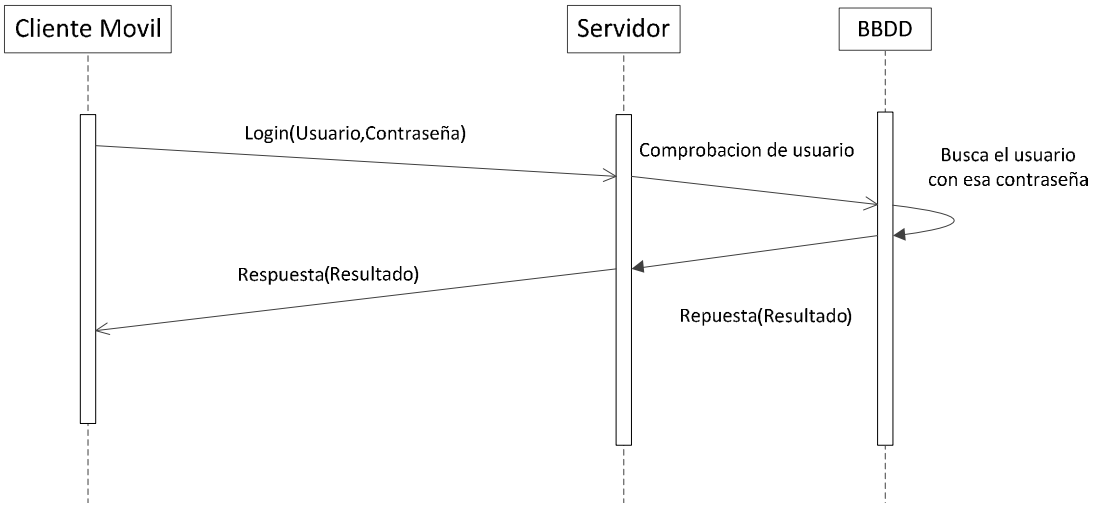


Ilustración 22: Diagrama de secuencia de Login.

3.5.5 Partida Multijugador

La figura detalla el flujo de mensajes entre servidor y las aplicaciones móviles.

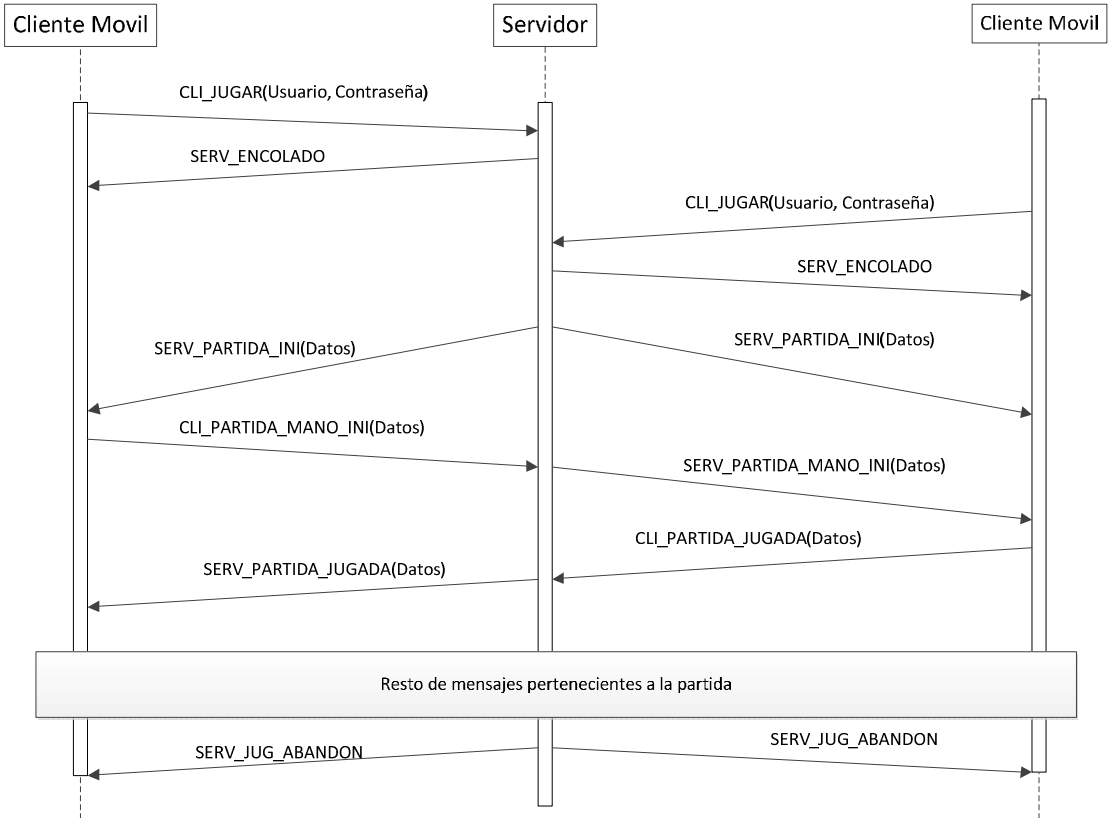


Ilustración 23: Diagrama de secuencia partida online.

4 DISEÑO DE LA APLICACIÓN

En este apartado se detalla el diseño del protocolo de transmisión entre el cliente y el servidor, la interfaz de usuario gráfica y el diseño del diagrama de clases previo a la implementación.

4.1 DIAGRAMAS DE DISEÑO DE LA APLICACIÓN

I) Diagrama de despliegue

En la siguiente figura se muestra la topología de la plataforma mediante los nodos que la integran. Estos nodos son elementos físicos separados con hardware y memoria propia. El nodo aplicación móvil contendrá los paquetes SW, componentes y clases necesarias para la ejecución de la aplicación del juego de cartas sobre el IDH. Dicho nodo se comunica a través de las facilidades OTA del dispositivo móvil (WiFi, GSM, GPRS, UMTS, etc.) con el componente de soporte a la aplicación alojado en servidor web implementado en el PFC ya mencionado cuya autora es Mercedes Sánchez Maroto [SAN13].

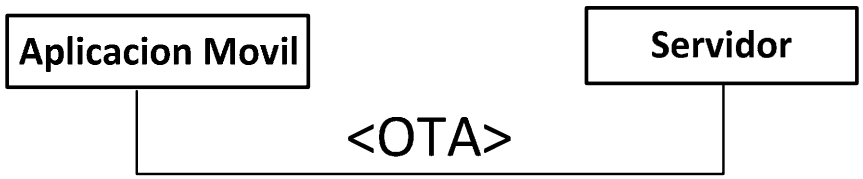


Ilustración 24: Diagrama de despliegue

II) Diagrama de Clases

Descompondremos la aplicación en dos gráficos, ya que la aplicación tiene dos "*activities*", como si fueran dos aplicaciones distintas, por un lado los menús y por otro lado el juego en sí.

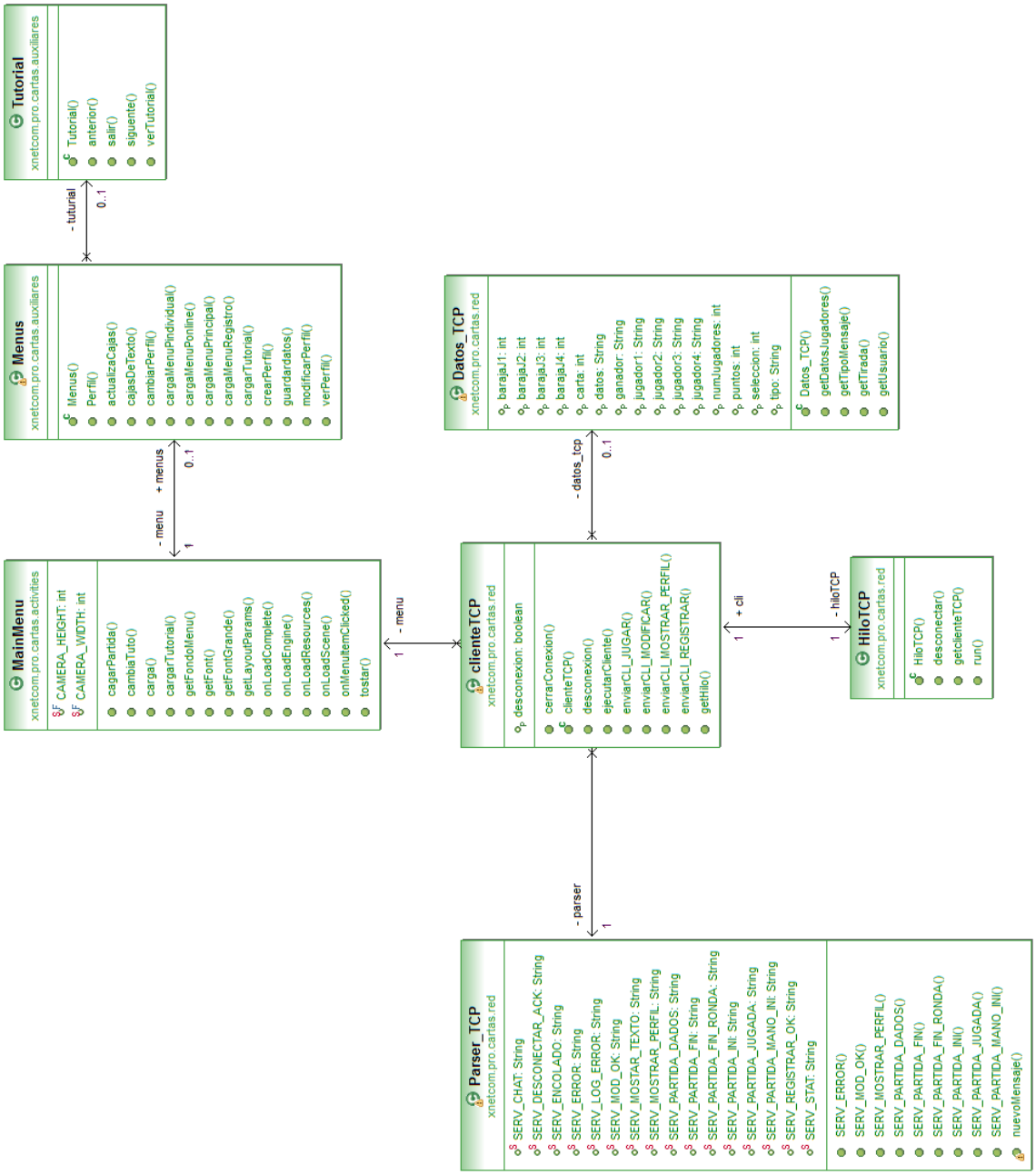


Ilustración 25: Diagrama de clases de los Menús.

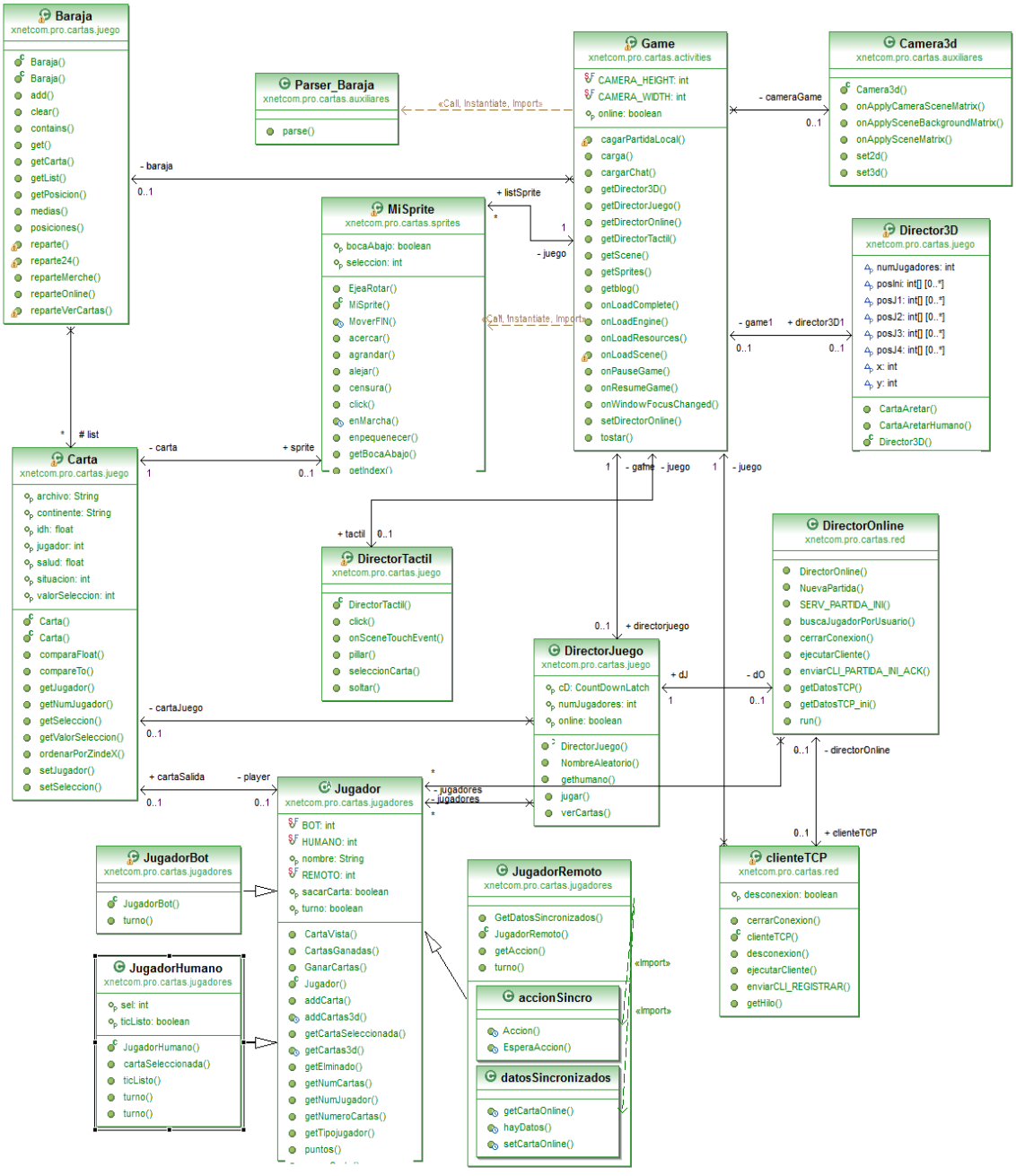


Ilustración 26: Diagrama de clases del Juego.

4.2 DISEÑO DEL PROTOCOLO CLIENTE MÓVIL-SERVIDOR WEB

La aplicación debe comunicarse con el servidor, para tal efecto se ha diseñado un protocolo entre el terminal móvil y el servidor escrito en el lenguaje XML, cuyos mensajes y usos se describen a continuación. Se han diseñado 8 tipos de mensajes, a modo de PDUs, para el intercambio entre cliente y servidor y 14 mensajes para el intercambio entre servidor y cliente.

Se considero la opción de nombrar los tipos de mansajes “<mensaje_ID>“ por un numero para disminuir el trafico, pero debido al bajísimo tráfico de datos que necesita el juego se decidió dejar con un nombre descriptivo ya que facilitaba mucho la comprensión del trafico de mensajes transmitidos si por ejemplo se guardaran en un log y para depuración.

Todos los mensajes constan de la siguiente estructura para facilitar su entendimiento y parséo, además los mensajes salientes del servidor empiezan siempre por “SERV_“ y los mensajes salientes de los clientes empiezan siempre por “CLI_“.

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>XXX</mensaje_ID>
    <datos>
    ...
    </datos>
</mensaje>
```

Etiquetas generales:

-TAG **<mensaje protocol_ver="1.0">:** Es la etiqueta que engloba todo el mensaje, contiene un atributo **protocol_ver="1.0"**, que sirve para identificar la versión de protocolo que se está utilizando.

-TAG **<mensaje_ID>**: Contiene el tipo de mensaje que se está enviando.

-TAG:**<datos>**: Contiene una serie de etiquetas con los datos propios del tipo de mensaje

A continuación detallaremos los diferentes tipos de mensajes: cliente a servidor y servidor a cliente.

I) Mensajes en dirección Cliente -> Servidor

I.1. PETICION DE INICIO DE UNA PARTIDA

```
<mensaje protocol_ver="1.0">
<mensaje_ID>CLI_JUGAR</mensaje_ID>
      <datos>
      <usuario>pedrito_123</usuario>
      <password>1234</password>
        <baraja>1.0</baraja>
      </datos>
</mensaje>
```

Este mensaje se manda cuando el usuario quiere empezar una partida multijugador.

-TAG **<mensaje_ID>:** Indica que el mensaje es un mensaje para el inicio de una nueva partida online.

-TAG **<usuario>** : El usuario que manda el mensaje

-TAG **<password>**: La contraseña de dicho usuario para comprobar que efectivamente es ese usuario.

-TAG **<baraja>**: Indica la versión de baraja que tiene el cliente instalado, para futuras versiones de la baraja.

I.2. PETICION DE DESCONEXION

```
<mensaje protocol_ver="1.0">
      <mensaje_ID>CLI_DESCONECTAR</mensaje_ID>
      <datos>
      <usuario>pedrito_123</usuario>
      <password>1234</password>
      </datos>
</mensaje>
```

Este mensaje se manda cuando el usuario quiere desconectarse de una partida.

-TAG **<mensaje_ID>:** Indica que el mensaje es un mensaje de desconexión de usuario para la partida que actualmente esta jugando.

-TAG **<usuario>** :El usuario que manda el mensaje

-TAG **<password>**: La contraseña de dicho usuario para comprobar que efectivamente es ese usuario.

I.3. PETICION DE REGISTRO

```
<mensaje protocol_ver="1.0">
        <mensaje_ID>CLI_REGISTRAR</mensaje_ID>
        <datos>
        <usuario>pedrito_123</usuario>
        <password>1234</password>
        <nombre>Pedro</nombre>
        <apellidos>RamirezPerez</apellidos>
        <pais>Spain</pais>
        <email>pedrito123@gmail.com</email>
        </datos>
</mensaje>
```

Mensaje enviado por el usuario para registrarse en el servidor, debe rellenar los campos de datos de un formulario en el terminal móvil

-TAG **<mensaje_ID>:** Indica que el mensaje es un mensaje de registro de un usuario.

-TAG **<usuario>** : El usuario que manda el mensaje

-TAG **<password>**: La contraseña de dicho usuario para comprobar que efectivamente es ese usuario.

-TAG **<apellidos>**: Contiene los apellidos del usuario

-TAG **<pais>**: El país del usuario

-TAG **<email>:** Indica el correo electrónico que servirá entre otras cosas recuperar la contraseña si se olvido

I.4. PETICION DE MODIFICAR PERFIL

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>CLI_MODIFICAR</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    <password>1234</password>
    <new_password>4321</new_password>
    <nombre>Pedro</nombre>
    <apellidos>RamirezPerez</apellidos>
      <pais>Spain</pais>
    <avatar>1254</avatar>
    </datos>
</mensaje>
```

Mensaje enviado por el usuario para modificar los datos personales asi como de la contraseña o el correo electrónico.

-TAG **<mensaje_ID>:** Indica que el mensaje es un mensaje de modificación de datos de un usuario.

-TAG **<usuario>** : El usuario que manda el mensaje, este usuario no podrá ser cambiado.

-TAG **<password>**: La contraseña de dicho usuario para comprobar que efectivamente es ese usuario. En caso de que el usuario quisiera cambiarla, aquí vendría la contraseña anterior y la nueva estaría dentro de la etiqueta new_password.

-TAG **<new_password>**: La nueva contraseña de dicho usuario.

-TAG **<apellidos>** : Contiene los apellidos del usuario si estos han cambiado respecto a los que están contenidos en la base de datos del servidor se actualiza la base de datos.

-TAG **<pais>**: El país del usuario

-TAG **<email>:** Indica el correo electrónico que servirá entre otras cosas recuperar la contraseña si se olvido

I.5. PETICION DE VER EL PERFIL Y ESTADISTICAS

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>CLI_MOSTAR_PERFIL</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    <password>1234</password>
    </datos>
</mensaje>
```

Mensaje enviado para comprobar el perfil y sus estadísticas del jugador, partidas jugadas, ganadas, perdidas, nivel, etc.

-TAG **<mensaje_ID>:** Indica que el cliente quiere recibir sus estadísticas

-TAG **<usuario>** : El usuario que manda el mensaje,

-TAG **<password>**: La contraseña de dicho usuario para comprobar que efectivamente es ese usuario.

I.6. MENSAJE DE ACK DE INICIO DE PARTIDA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>CLI_PARTIDA_INI_ACK</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    </datos>
</mensaje>
```

Indica al servidor que el cliente ya está listo para empezar la partida, una vez que todos los usuarios en esta partida mandan este mensaje se inicia la partida.

-TAG **<usuario>:** Usuario que manda el mensaje

I.7. MENSAJE DE INICIO DE MANO

```
<mensaje protocol_ver="1.0">
    <mensaje_ID> CLI_PARTIDA_MANO_INI</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    <carta>12</carta>
    <seleccion>1</seleccion>
    </datos>
</mensaje>
```

Este mensaje lo envía el jugador que empieza el turno indicando que carta y que parámetro es esta seleccionado.

-TAG **<usuario>:** Usuario que manda el mensaje.

-TAG **<carta>** El numero de la carta seleccionada por el usuario, cada numero de carta corresponde a un país,

-TAG **<seleccion>:** indica que su selección de dato.

I.8. MENSAJE DE SELECCIÓN DE CARTA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>CLI_PARTIDA_JUGADA</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    <carta>12</carta>
    <seleccion>1</seleccion>
    </datos>
</mensaje>
```

Es el mensaje que envía cualquier jugador para indicar su carta y su selección de dato después de que un jugador empezara la mano.

-TAG **<mensaje_ID>:**Indica el tipo de mensaje.

-TAG **<usuario>:** Usuario que manda el mensaje.

-TAG **<carta>** El numero de la carta seleccionada por el usuario, cada numero de carta corresponde a un país.

-TAG **<seleccion>:**indica que su selección.

II) Mensajes en dirección Servidor -> Cliente

II.1. MENSAJE DE USUARIO ENCOLADO

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_ENCOLADO</mensaje_ID>
</mensaje>
```

El servidor indica el cliente que ha sido encolado y que se le asignara una partida cuando se encuentre otro jugador.

II.2. MENSAJE DE ACK DE DESCONEXIÓN DEL CLIENTE

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_DESCONECTAR_ACK</mensaje_ID>
</mensaje>
```

El servidor mandara un ack al usuario cuando este le pida desconexión.

II.3. MENSAJE DE CONFIRMACIÓN EXITOSA DE REGISTRO

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_REGISTRAR_OK</mensaje_ID>
</mensaje>
```

El servidor mandara este mensaje una vez el registro sea exitoso, en caso contrario mandara un mensaje de error con un texto de descripción.

II.4. MENSAJE DE ERROR

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_ERROR</mensaje_ID>
    <datos>
    <texto>el mensaje con el error</texto>
    </datos>
</mensaje>
```

Mensaje que informa que un jugador ha dejado la partida.

-TAG **<texto>**: Texto descriptivo del error, este mensaje será mostrado al usuario.

II.5. MENSAJE DE ABANDONO DE JUGADOR

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_JUG_ABANDON</mensaje_ID>
    <datos>
    <usuario>pablo_23</usuario>
    </datos>
</mensaje>
```

Este mensaje lo manda el servidor a los demás jugadores de la partida, cuando un jugador abandona la partida, si quedan jugadores suficientes se continua la partida con los jugadores restantes.

-TAG **<usuario>:** Jugador que deja la partida.

II.6. MENSAJE CON LOS DATOS DEL PERFIL

```
<mensaje protocol_ver="1.0">
<mensaje_ID>SERV_MOSTAR_PERFIL</mensaje_ID>
      <datos>
      <usuario>pedrito_123</usuario>:
      <nombre>Pedro</nombre>
      <apellidos>RamirezPerez</apellidos>
      <pais>Spain</pais>
      <nivel>54</nivel>
      <jugadas>85</jugadas>
      <ganadas>34</ganadas>
      <puntos>84574</puntos>
      </datos>
</mensaje>
```

Este mensaje lo manda el servidor cuando recibe una petición de visualización de datos del perfil.

-TAG **<usuario>**:El usuario del perfil

-TAG **<nombre>**Nombre del usuario

-TAG **<apellidos>**Apellidos del usuario

-TAG **<pais>**País del usuario

-TAG **<nivel>**Nivel del usuario, en base al la cantidad de partidas jugadas y ganadas, se utilizará este para tratar de enfrentar a jugadores del mismo nivel.

-TAG **<jugadas>**Partidas jugadas

-TAG **<ganadas>**Partidas ganadas

-TAG **<puntos>**Puntos adquiridos

II.7. MENSAJE DE CONFIRMACIÓN DE MODIFICACIÓN DE PERFIL

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_MOD_OK</mensaje_ID>
</mensaje>
```

Este mensaje confirma al usuario que su perfil (nombre, apellidos, país) se ha modificado correctamente.

II.8. MENSAJE DE INFORMACION

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_MOSTAR_TEXTO</mensaje_ID>
    <datos>
    <texto>el mensaje informativo</texto>
    </datos>
</mensaje>
```

Este mensaje contiene cualquier información que el servidor quiera mandar en cualquier momento a los clientes, esta información se mostrara al usuario en un cuadro de texto emergente.

II.9. MENSAJE DE INICIALIZACION DE PARTIDA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_INI </mensaje_ID>
    <datos>
    <num_jugadores>2</num_jugadores>
    <jugador1>
    <usuario>Marco_7</usuario>
    <nivel>24</nivel>
<baraja>
3:45:56:173:23:28:67:36:96:24:76:43:75:27:74:
47:28:12:79:12:86:124:175:145:125:123:136
</baraja>
    </jugador1>
    <jugador2>
    <usuario>pedrito_123</usuario>
    <nivel>23</nivel>
<baraja>
5:95:52:179:33:21:77:86:96:44:77:43:75:27:74:47:28:12:7
9:12:85:12:15:145:25:163:131
</baraja>
    </jugador2>
    </datos>
</mensaje>
```

Este mensaje lo envía el servidor a todos los jugadores de la partida. Contiene todos los datos necesarios para inicializar la partida, como son los jugadores que intervienen y sus cartas, puede contener hasta cuatro jugadores que es el máximo para una partida.

-TAG **<num_jugadores>**: Indica el número de jugadores

-TAG **<jugador1>**: Indica que dentro de este tag se contiene la información del jugador 1, de la misma forma que se haría con los jugadores 2, 3 y 4.

-TAG **<usuario>**: Indica el Nick del jugador

-TAG **<baraja>**:Indica las cartas que posee dicho jugador.

-TAG **<nivel>**:Indica el nivel del jugador.

II.10. MENSAJE DE TIRADA DE DADOS

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_DADOS</mensaje_ID>
    <datos>
    <turno>pedrito_123</turno>
    </datos>
</mensaje>
```

Antes de comenzar la partida hay una tirada de dados para determinar qué jugador empieza la mano. El servidor determina quién es ese jugador y envía este mensaje a todos los jugadores, la partida empieza en ese momento. El motivo para enviar esta información por separado y no en el mensaje de Serv_partida_ini, es dejar tiempo a que se monten las cartas con las texturas y después comenzar la partida.

-TAG**<turno>**: Indica que jugador será el que empiece la primera mano.

II.11. MENSAJE DE COMIENZO DE MANO

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_MANO_INI</mensaje_ID>
    <datos>
    <usuario>pedrito_123</usuario>
    <carta>12</carta>
    <seleccion>1</seleccion>
    </datos>
</mensaje>
```

Una vez el servidor ha recibido el mensaje de **CLI_PARTIDA_MANO_INI** del cliente que inicia la mano, este retransmite este mensaje a los demás clientes de esta partida.

-TAG **<usuario>**: Usuario que inicio la mano.

-TAG **<carta>** : El numero de la carta seleccionada por el usuario, cada numero de carta corresponde a un país,

-TAG **<seleccion>:** Indica su selección

II.12. MENSAJE DE SELECCIÓN DE CARTA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_JUGADA</mensaje_ID>
    <datos>
    <usuario>Jorgito_487</usuario>
    <carta>15</carta>
    <seleccion>1</seleccion>
    </datos>
</mensaje>
```

Cuando un jugador que no es mano le manda un mensaje al servidor **CLI_PARTIDA_JUGADA** este retransmite a los demás jugadores el mensaje **SERV_PARTIDA_JUGADA** con la misma información.

-TAG **<usuario>**:Usuario que hizo la selección.

-TAG **<carta>** : El numero de la carta seleccionada por el usuario, cada numero de carta corresponde a un país,

-TAG **<seleccion>:** Indica su selección

II.13. MENSAJE DE FIN DE RONDA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_FIN_RONDA</mensaje_ID>
    <datos>
    <num_jugadores>2</num_jugadores>
    <ganador>Jorgito_321</ganador>
    <puntos>46</puntos>
    <jugador1>
    <usuario>Jorgito_321</usuario>
    <carta>24</carta>
    <seleccion>3</seleccion>
    </jugador1>
    <jugador2>
    <usuario>pedrito_123</usuario>
    <carta>23</carta>
<seleccion>3</seleccion>
    </jugador2>
    </datos>
</mensaje>
```

Al finalizar una ronda también llamada mano, Se muestra un mensaje a todos los jugadores en el que se encuentra las cartas que cada jugador ha sacado y cual es el ganador la información se saca de este mensaje que el servidor manda a los jugadores en esta ronda, el ganador será el jugador que comience la siguiente ronda.

-TAG **<ganador>**: El jugador que gano la mano

-TAG **<puntos>**: Los puntos que ha ganado en esta mano

-TAG **<seleccion>:** Indica su selección

II.14. MENSAJE DE FIN DE PARTIDA

```
<mensaje protocol_ver="1.0">
    <mensaje_ID>SERV_PARTIDA_FIN</mensaje_ID>
    <datos>
    <num_jugadores>2</num_jugadores>
    <ganador>Jorgito_321</ganador>
    <jugador1>
    <usuario>Jorgito_321</usuario>
    <puntos>1233</puntos>
    </jugador1>
    <jugador2>
    <usuario>pedrito_123</usuario>
    <puntos>334</puntos>
    </jugador2>
    </datos>
</mensaje>
```

Es prácticamente igual a la anterior, pero esta se manda al final de la partida y contiene los puntos finales de los jugadores y determina quien ganó la partida.

4.3 DISEÑO E IMPLEMENTACION DE LA INTERFAZ GRAFICA

Desde el punto de vista de la ingeniería de software, la Interfaz de Usuario Gráfica (GUI) tiene un papel fundamental puesto que determina la forma mediante la cual el usuario interactúa con la aplicación. Los criterios de diseño de la GUI desarrollada para la aplicación objeto de este PFC asumen desde el principio la necesidad de poder ser utilizada fácilmente por un niño de al menos 9 años. Por este motivo, es esencial que su uso sea fácil e intuitivo, siendo congruente con los principios fundamentales de la ingeniería de usabilidad. Un ejemplo de las consideraciones de diseño tenidas en cuenta ha sido eliminar en su mayor parte el uso de menús durante el juego, para que el usuario tuviera la sensación de tener en sus manos una baraja, mediante el uso de metáforas del mundo real para la implementación de la dinámica del juego, tales como el toque de las cartas mediante pantalla táctil del dispositivo para arrastrarlas, pasarlas, soltando, tocando, etc.

4.3.1 Consideraciones de Usabilidad

La usabilidad puede describirse como el atributo que mide la facilidad de uso que tiene una interfaz, para conseguir una interfaz gráfica con un gran grado de usabilidad hay que cumplir una serie de premisas:

- **Facilidad de Aprendizaje:** facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema. Está relacionada con la predictibilidad, congruencia de la interfaz, su formación previa y la familiaridad con el sistema.
- **Facilidad de Uso:** facilidad con la que el usuario utiliza de la herramienta, con menos pasos o más naturalidad. Tiene que ver con la eficacia y eficiencia de la herramienta.
- **Flexibilidad:** tiene que ver con la variedad de posibilidades que el usuario tiene para interactuarcon el sistema. También abarca la optimización entre el usuario y el sistema.
- **Robustez:** es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

En este se han seguido estas condiciones de usabilidad, se ha tratado de que la interactuación del usuario con la aplicación fuera lo más intuitivo posible, tratando en todo lo posible que el usuario utilice la aplicación de la misma forma que lo haría si estuviese en una mesa y una baraja real en las manos.

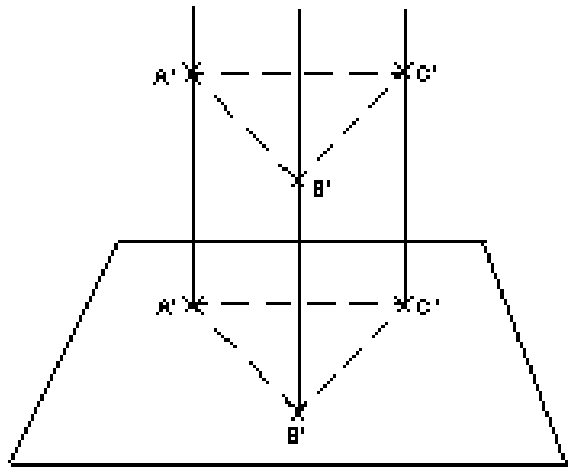
4.3.2 Diseño de la perspectiva

Toda la interfaz gráfica se ha diseñado a bajo nivel, debido a la inexistencia de ningún editor gráfico para el frameworkAndEngine,algo similar seria la clase Canvas de Java. Este método aunque es más lento permite un control total de los recursos y permite gestionar entre otras cosas el número de texturas en memoria así como de las superficies táctiles que están siendo "escuchadas" por el sistema.

AndEngine es un Framework para el diseño de videojuegos en 2D, la perspectiva que utiliza es la proyección cilíndrica ortogonal, no obstante gracias al uso configurable de Open GL se modificó dicha perspectiva para utilizar la proyección central o cónica, también se implementó un nuevo sistema de control de superficies táctiles y un nuevo eje Z para la profundidad, ya que al estar diseñado para 2D no se contaba con un eje con la profundidad. De esta forma obtenemos una aplicación en 3D.

A continuación se muestra un gráfico representativo de las dos perspectivas.

Proyección ortogonal: Es aquella cuyas rectas proyectantes auxiliares son perpendiculares al plano de proyección (o a la recta de proyección), estableciéndose una relación entre todos los puntos del elemento proyectante con los proyectados.

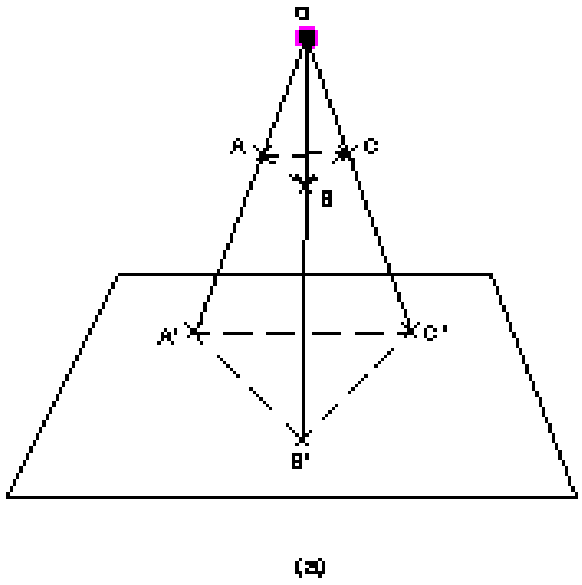


Como puede verse los rayos van rectos por lo que el objeto se verá igual de grande dando igual la distancia al plano.

Ilustración 27: Proyección Ortogonal.

Proyección cónica: en este sistema de representación el haz de rayos proyectantes confluye en un punto (el ojo del observador), proyectándose la imagen en un plano auxiliar situado entre el objeto a representar el punto de vista.

Es el sistema de representación que ayuda a reproducir (normalmente en un plano) las imágenes del modo más fiel, con un resultado muy similar a como lo percibimos realmente.



Como puede verse los rayos confluyen en un foco, con lo que al estar el objeto más cerca del foco se verá más grande.

Ilustración 28: Proyección Cónica.

A continuación se puede ver una comparación real de las dos proyecciones en el juego. La proyección ortogonal no ofrece al usuario una visión clara de que la carta está rotando y dándose la vuelta, más bien parece que se encoge desde los lados.

Ilustración 29: Implementación de la proyección Ortogonal.

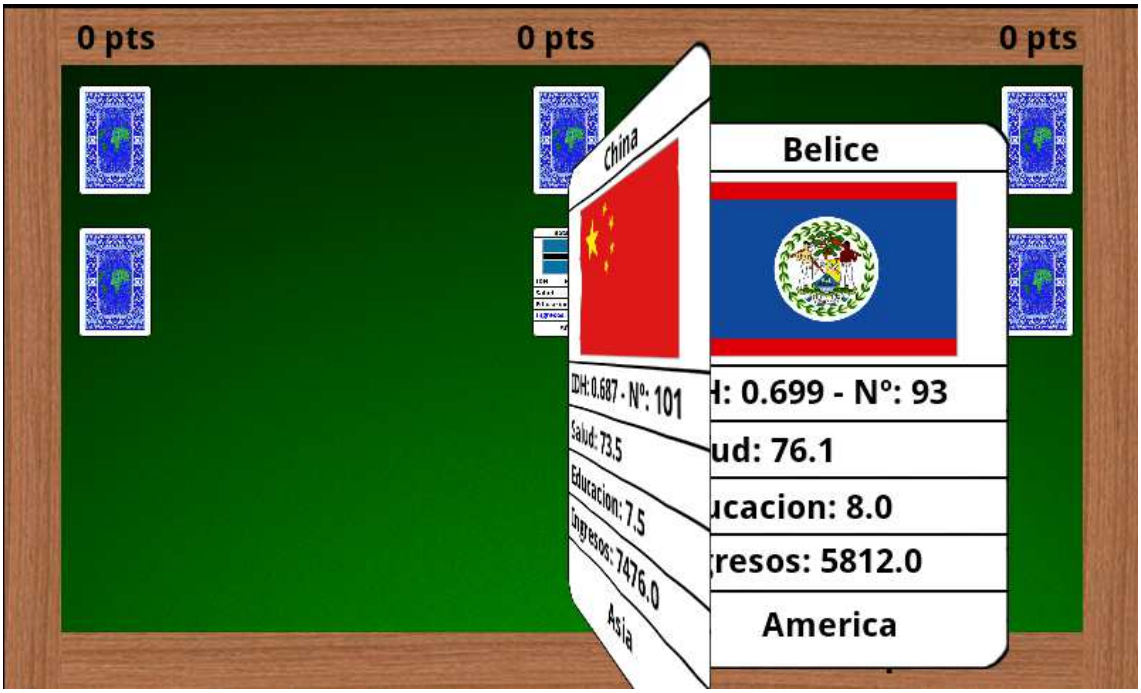


Ilustración 30: Implementación de la proyección Cónica.

Con este código en la cámara se cambia la perspectiva:

```
privatevoid setFrustum(GL10 pGL)
	{

	// setea el campo de vision a 60 grados
	float fov_grados = 60;
	float fov_radianes = fov_grados / 180 * (float)Math.PI;

	// configura el aspecto y la distancia de la camara
	float aspecto = this.getWidth() / this.getHeight();
	float camZ = this.getHeight()/2 / (float)Math.tan(fov_radianes/2);

	// configura la  proyeccion
		GLHelper.setProjectionIdentityMatrix(pGL);
		GLU.gluPerspective(pGL, fov_grados, aspecto, camZ/10,
camZ*10);

	// configura la vista
		GLU.gluLookAt(pGL,0,0,camZ,0,0,    0,  0,   1, 0);

		pGL.glScalef(1,-1,1);
	pGL.glTranslatef(-800/2,-480/2,0); // pone el origen de coordenadas

	}
```

Ilustración 31: Código para paso a 3D.

4.3.3 Implementación de la interfaz grafica

En este apartado se mostraran capturas del juego y se explicará su funcionalidad.

Pantalla de bienvenida: Mientras los pocos segundos que se tarda en cargar las texturas del los menús, se muestra una pantalla de bienvenida, el motivo central de esta pantalla es un mapamundi rodeado de niños de diferentes etnias cogidos de la mano rodeando el mundo, que en mi opinión recoge en un cuadro la motivación del juego.

Ilustración 32: Pantalla de bienvenida.

Pantalla de menú Principal: Muestra las opciones principales que tiene el juego: jugar de forma individual, Jugar en modo Multijugador por internet, opciones del perfil, ver las cartas, un tutorial y salir de la aplicación.

Ilustración 33: Pantalla de menú principal

Pantalla de Partida individual: Permite elegir el número de contrincantes he ir atrás.

Ilustración 34: Pantalla partida individual

Pantalla de menú de perfil: al presionar en el menú principal el botón perfil, se nos muestra este segundo menú en el que tenemos las diferentes opciones sobre el perfil: ver perfil, Modificar perfil, Cambiar perfil, Crear perfil y salir al menú principal.

Ilustración 35: Pantalla menú de perfil

Pantalla ver perfil: Es esta pantalla el usuario puede ver su perfil:

Ilustración 36: Pantalla de ver perfil

Pantalla de creación de perfil: Es esta pantalla el usuario introduce sus datos por primera vez, estos datos se mandan al servidor y si el proceso ha sido exitoso quedara registrado, y la aplicación móvil guardará en memoria el usuario y contraseña para posteriores partidas.

Ilustración 37: Pantalla de creación de perfil

Pantalla de modificación de perfil: En esta pantalla los cuadros de texto son editables y se dispone de un votos de guardar, que mandara al servidor los cambios realizados en el perfil.

Ilustración 38: Pantalla de modificación de perfil

Pantalla de cambio de usuario: En esta pantalla se muestra una ventana de dialogo de Android en el que se pide el nombre de usuario y contraseña,

Ilustración 39: Pantalla de cambio de usuario

Pantalla tutorial: en esta pantalla una serie de diálogos mostrara el uso de la aplicación a los usuarios. Consta de tres flechas de navegación, siguiente, anterior y salir al menú.

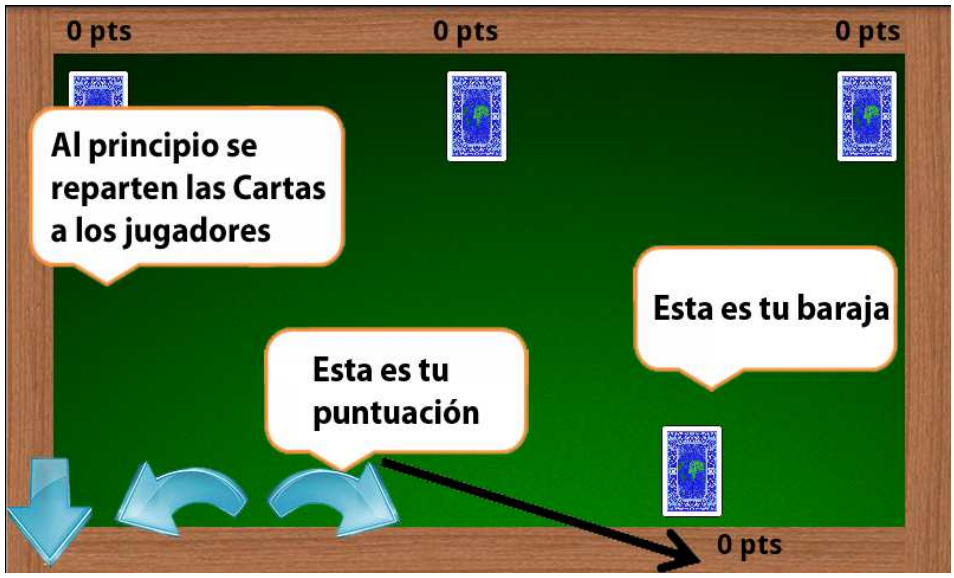


Ilustración 40: Pantalla tutorial.

Pantalla de carga: Esta pantalla aparece cada vez que se va a iniciar una partida, debido a que tiene que cargar muchas texturas de todas las banderas de todos los países, dependiendo del dispositivo puede tardar unos segundos, de modo que se muestra esta figura que gira según carga las texturas.

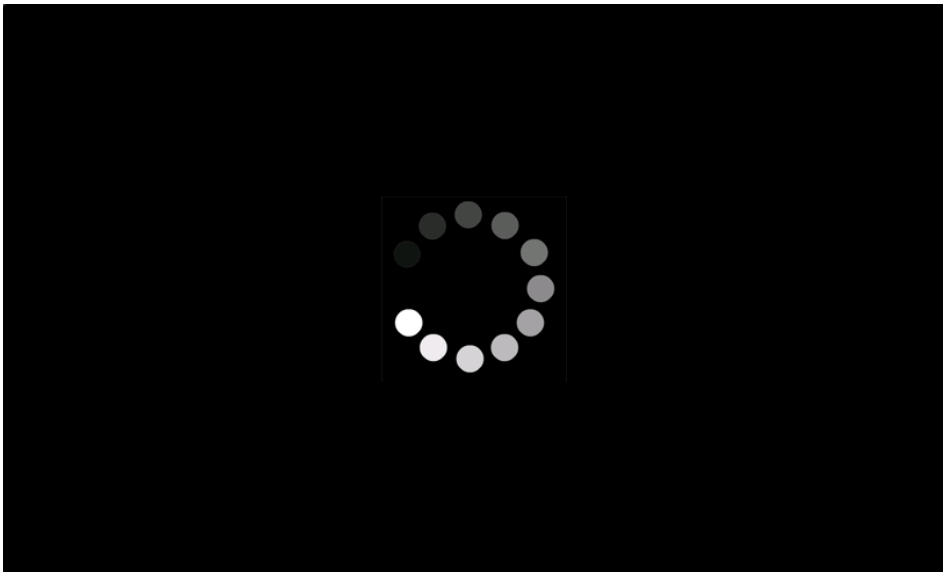


Ilustración 41: Pantalla de carga

Pantalla ver cartas: En esta pantalla el usuario no juega contra ningún adversario, dispone de un mazo con las 187 cartas disponibles en el juego, esta pantalla sirve para que los usuarios estudien los parámetros de todos los países, de esta forma los jugadores profundizar en sus conocimientos de los datos para ser mejores jugadores.

Ilustración 42: Pantalla de partida ver cartas

Pantalla de juego: En esta pantalla se desarrolla el juego, existen cuatro posiciones para un máximo de cuatro jugadores, la posición del usuario siempre es la misma, que es la que más cerca esta de él, abajo a la derecha

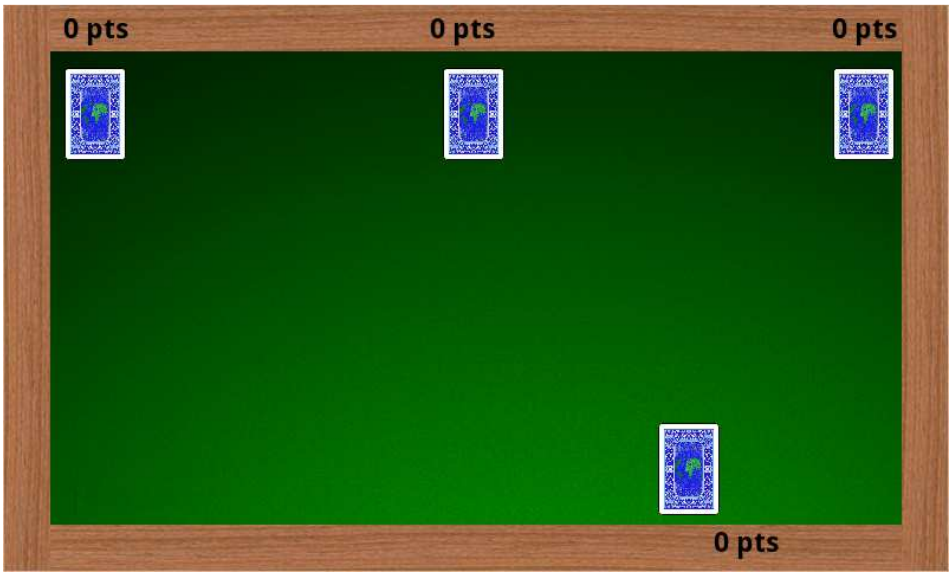


Ilustración 43: Pantalla de juego

Pantalla de resultados: Esta pantalla muestra los resultados de la mano indicando que cartas tenían los oponentes sus valores y cual es el ganador. Cuando termina la partida se muestra esta pantalla con los puntos totales del juego y el Ganador.

Ilustración 44: Pantalla de resultados

5 IMPLEMENTACIÓN Y PRUEBAS

5.1 HERRAMIENTAS DE DESARROLLO DE LA APLICACIÓN

El desarrollo de esta aplicación se ha realizado de forma incremental, añadiendo poco a poco nuevas característica y funcionalidades.

El uso del framework AndEngine no dejó otra opción, ya que no existe documentación oficial y poca no oficial de cómo desarrollas aplicaciones con dicho framework, el único material oficial del que se dispone, es un conjunto de aplicaciones de ejemplo que solo cubren una pequeña parte del desarrollo de una aplicación como la creada en este PFC. Haciendo necesario la investigación y la lectura de muchos foros para continuar el desarrollo, así como del uso intensivo de la técnica comúnmente llamada “Prueba y error”, con la consecuente pérdida de tiempo.

Durante el desarrollo de esta aplicación se abandonó el uso de AndEngine 2 veces y se probaron diferente frameworks gratuitos con mayor documentación. Pero el nivel de madurez y compatibilidad de la mayoría era insuficiente habiéndose abandonado el desarrollo de ellos en muchos casos.

Como se deseaba una buena apariencia estética y un buen rendimiento con dispositivo de gama media-baja no se consideró el uso de programar la aplicación directamente en Android y se decidió, a pesar de los inconvenientes la realización de la aplicación con el framework AndEngine.

A continuación se describen las herramientas que se han utilizado para el desarrollo de la aplicación.

I) Eclipse

Eclipse es un entorno de desarrollo integrado IDE (Integrated Development Environment). Un IDE consiste en un editor de código, un compilador, un depurador y un constructor deinterfaz gráfica (GUI).

Fue creado originalmente por IBM, actualmente sigue desarrollándose por la Fundación Eclipse y es una aplicación de software libre.

La principal virtud de este entorno es que facilita enormemente las tareas de programación depuración y compilación, aunque es utilizado principalmente para el lenguaje Java es compatible con otros lenguajes de programación como C++, PHP o Python.

Está fuertemente apoyado por plugins, y dispone de repositorios propios para ello, de forma que para añadirle nuevas funcionalidades solo hay que descargárselos. Pudiendo hacerlo desde el propio programa. Además tiene de serie la posibilidad de hacer pruebas unitarias con JUnit y control de versiones con CVS.

Para programar en Android hay que instalar el plug-in ADT (Android Development Tools plugin). Esta utilidad nos permitirá programar para Android utilizando las librerías de Android y podremos compilar las aplicaciones y ejecutarlas en un dispositivo real conectado por USB, esto es mucho mejor que utilizar el emulador que viene de serie con el Android SDK ya que es muy lento incluso en máquinas rápidas ya que tiene que emular el procesador RISC que disponen los dispositivos a emular.

II) Android SDK

El SDK (Software Development Kit) de Android está compuesto por un depurador de código, bibliotecas, un simulador de teléfono, ejemplos de código y tutoriales. Nos permite elegir para que versión de Android se va a programar y elegir un emulador que ejecute la versión determinada de Android para hacer las pruebas, el emulador permite emular diferentes dispositivos existentes como el Nexus 4, pero también podemos configurar nuestro propio dispositivo personalizado, configurando la RAM, la versión de Android y el tamaño de la pantalla, en la siguiente figura podemos ver el emulador de Android.

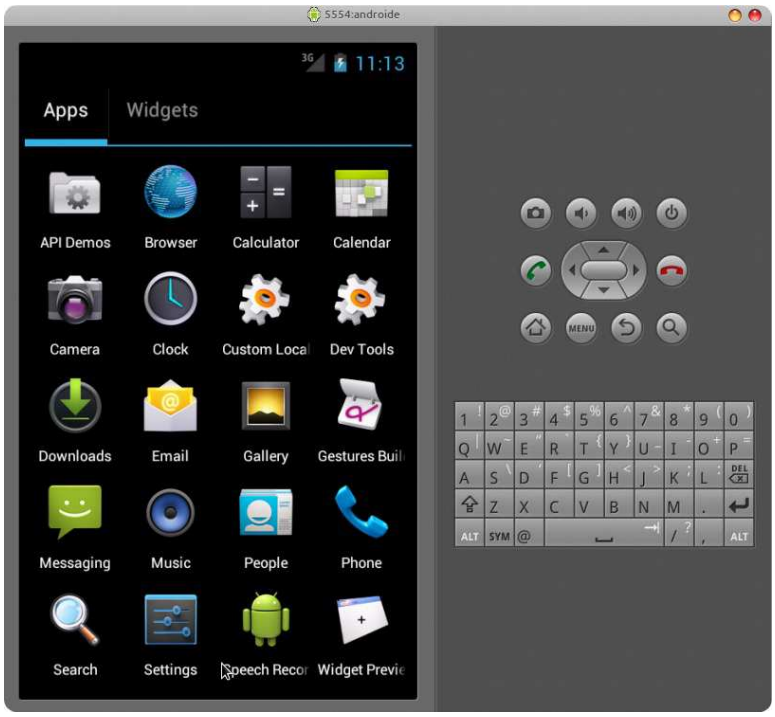


Ilustración 45: Emulador de Android.

III) Notepad ++

Es un editor de texto, con licencia GNU, lo que significa que es gratuito y su código fuente está disponible por si se quiere modificar.

Es compatible con muchos lenguajes de programación y mediante llamadas a un compilador se puede compilar y ejecutar el software que se cree con este editor.

En este proyecto se ha utilizado para el diseño de los mensajes XML que se intercambian el cliente y el servidor y para crear un archivo XML que contiene la información de las cartas de los países.

IV) Photoshop

Es un programa de edición grafica que permite mediante la utilización de múltiples capas la composición de gráficos a partir de pequeños fragmentos, permite aplicar multitud de efectos y filtros a las imágenes, así como acciones básicas como recortar modificar el tamaño de los gráficos, etc.

La interfaz gráfica prácticamente ha sido creada con esta herramienta.

5.2 IMPLEMENTACIÓN DE LAS CARTAS

En el desarrollo de las cartas se ha tratado en todo momento de que su diseño sea fácil y familiar a los jugadores. También se ha intentado en todo momento en la maximizar la agilidad del sistema y del tamaño dela aplicación, las cartas se "montan" cuando se carga el juego a partir de texturas por separado, por ejemplo la textura del borde de la carta está presente en todas las cartas pero solo ocupa espacio en memoria una vez y las cartas comparten la dirección de memoria donde está almacenado el único prototipo.

El resto de datos se carga de un archivo XML en donde se le dice a la aplicación todos los datos necesarios para montar las cartas, abajo se encuentra una porción del código XML. El código XML contiene la versión de la baraja, este dato es enviado al servidor cuando se desea iniciar una partida multijugador, en caso de que su versión este obsoleta, sea visará al usuario para que actualiza la aplicación.

Esta técnica permite la actualización fácil de los datos humanitarios de las cartas con solo cambiar el archivo XML.

Código XML con el que se montan las cartas:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <baraja id="1.0">
  - <pais id="Noruega">
      <pidh>1</pidh>
      <idh>0.943</idh>
      <salud>81.1</salud>
      <educacion>12.6 </educacion>
      <ingresos>47557</ingresos>
      <continente>Europa</continente>
      <archivo>noruega</archivo>
      <numero>1</numero>
    </pais>
  - <pais id="Australia">
      <pidh>2</pidh>
      <idh>0.929</idh>
      <salud>81.9</salud>
      <educacion>12.0 </educacion>
      <ingresos>34431</ingresos>
      <continente>Oceania</continente>
      <archivo>australia</archivo>
      <numero>2</numero>
    </pais>
  - <pais id="Países Bajos">
      <pidh>3</pidh>
      <idh>0.910</idh>
```

Ilustración 46: Código de las cartas en XML.

Una vez se ha leído los datos necesarios para el montaje de una carta se crea el "Sprite Carta" que contiene el borde base y al que se le van adjuntando en posiciones determinadas la textura de la bandera del país, el dorso, y de los datos de texto correspondientes. Cada dato de la carta cuenta con su propia superficie táctil para el sistema, de modo que cuando se desee seleccionar un dato en cuestión de la carta, bastará con tocar dicho valor y el color del texto seleccionado se volverá azul, cuando se toque otro dato, el anterior dato seleccionado volverá a ser negro y el nuevo dato se pondrá azul.

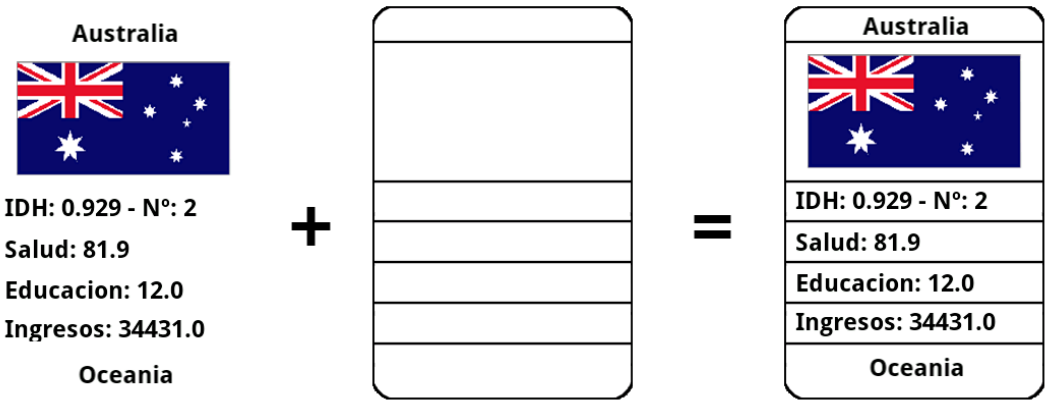


Ilustración 47: Construcción de la carta

Dorso de las cartas: El dorso de las cartas ha sido diseñado con un aspecto clásico con un mapamundi en el centro como motivo central. Al igual que como el borde solo se carga una vez en memoria, es volteado y adjuntado al resto de texturas de la carta con una opción que la hace invisible cuando esta dado la vuelta para evitar problemas de intrusión de texturas y sincronismo por perdida de precisión del Z-Buffer de Open GL.Ya que ambos gráficos son coplanarios, es decir comparten la misma coordenada Z.

Con las demás texturas de la carta se les aplica la misma técnica.

Ilustración 48: Dorso de las cartas.

Ya se ha hablado antes como se ha optimizado el uso de memoria con las cartas pero hay otro factor muy importante en esta aplicación y es que se cuenta con 187 cartas, cada carta con 5 superficies táctiles independientes, cuatro para los datos y una global para la carta, lo que nos da un total de 935 superficies táctiles solamente de las cartas.

Además del uso de memoria gráfica de todas las banderas, dado que realmente solo se puede ver dos cartas de los contrincantes y 3 cartas del usuario cuando se está pasando una carta de un lado a otro. Se ha optado por un modo de attach y detach automático. Según se van pasando las cartas de un lado a otro detecta a qué lado se están pasando las cartas y cuando hay 3 apiladas en un lado aplica un detach a la tercera por arriba, por el otro lado cuando detecta que solo queda dos cartas visibles y se realiza un attach de una tercera, de este modo cuando se toque una carta, el array de superficies táctiles es mucho más pequeño, la respuesta táctil se agiliza y las texturas salen de la memoria gráfica a la del sistema que es mucho más abundante. Con esta técnica se consigue que las cartas que están por debajo no consuman recursos. Una técnica parecida se aplica a las cartas de los adversarios

5.3IMPLEMENTACIÓN DE LA APLICACIÓN

En este apartado se nombran las clases principales y se mostrara las partes de su código más representativo de la función de estas clases.

Clase Game

Es la clase principal del juego, extiende de la clase BaseGameActivity.

Es la que se ejecuta cuando se va a realizar una partida, se encarga de cargar todas las texturas. También se va pasando a las demás clases como contexto por si necesitan usar

alguno de los métodos que hereda esta clase, si necesitara comunicarse con otra clase, se realiza a través de ella.

Es siguiente código muestra la carga de las texturas de las banderas

```
BTAflags = new BitmapTextureAtlas[numero];

for (int i = 0; i < numero; i++) {
	this.BTAflags[i] = new BitmapTextureAtlas(256, 1024,
		TextureOptions.BILINEAR_PREMULTIPLYALPHA);
}
this.TRpaises = new TextureRegion[baraja.getList().size()];

BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/flags/");

int i = 0;
int j = 0;
int h = 0;

for (Iterator<Carta> it = baraja.getList().iterator(); it.hasNext(); i++,
h++) {
	Carta carta = (Carta) it.next();
	if (i == 8) {
		i = 0;
		j++;
	}
	TRpaises[h] = BitmapTextureAtlasTextureRegionFactory
		.createFromAsset(this.BTAflags[j],
		this, carta.getArchivo()+ ".png", 0, i * 128);
		Log.i("cargando", carta.getArchivo() + ".png");
}

SoundFactory.setAssetBasePath("mfx/");
	try {
		this.Sonido_Dados = SoundFactory.createSoundFromAsset(
				this.mEngine.getSoundManager(), this,
				"dados.mp3");

		} catch (final IOException e) {
			Debug.e(e);
		}
```

Una ver termine de cargar todas las texturas, los sonidos. Iniciara la partida, para ello creara y después iniciara una instancia de la clase DirectorJuego que es la encargada de gestionar la partida. En la siguiente porción de código se muestra como inicia una partida.

```
directorjuego = new DirectorJuego(this, dado, escenaJuego);
this.mEngine.setScene(escenaJuego);
directorjuego.nuevaPartida(numJugadores, online, baraja);
```

Clase DirectorJuego

Esta clase se encarga de todas las gestiones de la partida, contiene a los jugadores y se encarga de inicializarlos ya sean JugadorBot, JugadorHumano o JugadorOnline, se communica con el Director3D para las transiciones y efectos.

Se encarga de llevar el ritmo de la partida avisando a cada jugador cuando le llega su turno y que debe hacer. A continuación se puede ver parte del código que trata los turnos, esta tarea se realiza en un bucle que terminara cuando se acaben las rondas.

```
CD = new CountDownLatch(1);
jugadores[turno].turno(true, null);// turnodeljugador

try {
	CD.await();// espera a quetermine el truno
} catch (InterruptedException e) {
	e.printStackTrace();
}

CartaJuego = jugadores[turno].getCartaSeleccionada();//
cartaseleccionadapor el jugador
CD = new CountDownLatch(jugadores.length - 1);
	for (int i = 0; i <jugadores.length; i++) {
		if (i != turno)
			jugadores[i].turno(false, CartaJuego);
	}
try {
	CD.await();
} catch (InterruptedException e) {
	e.printStackTrace();
}
```

Clase Director3D

Esta clase se encarga de los efectos y transiciones, es este trozo de código pertenece a una función que reparte inicialmente las cartas desde el mazo a la posición de cada jugador.

```
for(Iterator<Carta>it=baraja.getList().iterator();it.hasNext();z++,i++){

Carta carta = (Carta)it.next();
MiSprite sprt =carta.getSprite();
int pos=carta.getNumJugador();

if(i>(baraja.getList().size()-numJugadores)){
sprt.registerEntityModifier(new MoverX(1f,sprt.getX(),posJugador(pos)[x]));
sprt.registerEntityModifier(new MoverY(1f,sprt.getY(),posJugador(pos)[y]));

try {
	Thread.currentThread().sleep(300);
} catch (InterruptedException e1) {
	e1.printStackTrace();
}
}
```

La siguiente figura muestra la función que se encarga de mover una carta a un punto de la pantalla y darle la vuelta, también cambia el Z-index de la carta para que este en primer plano y la registra en la escena si no lo estaba para ahorrar memoria.

```
publicvoid sacarCarta(Carta carta){
	carta.setSituacion(1);
	MiSprite sprt=carta.getSprite();
	if(!sprt.hasParent())sprt.registrar();
	sprt.setZIndex(sprt.getZIndex()+150);
	game.getScene().sortChildren();
	sprt.setVisible(true);
sprt.registerEntityModifier(newMoverY(1f,sprt.getY(),sprt.getY()+100));
	sprt.verCarta(2f);
}
```

Clase DirectorTactil.

Debido a que el Framework AndEngine daba problemas en cuanto a las superficies táctiles, se optó por la creación de una nueva clase que implementara la interfaz IOnSceneTouchListener y estuviera hecha a medida para la aplicación. A continuación podemos observar un trozo de código que se encarga de seccionar la carta tocada

```
publicvoid seleccionCarta(TouchEvent pSceneTouchEvent){
	if(juego.getTic().contains(pSceneTouchEvent.getX(),pSceneTouchEvent.
	getY())&&juego.getTic().isVisible()){
		juego.getTic().onAreaTouched(pSceneTouchEvent, 0,0);
		return;
	}
	if(juego.getblog().contains(pSceneTouchEvent.getX(),SceneTouchEvent.
	getY())&&juego.getblog().isVisible()){
		juego.getblog().onAreaTouched(pSceneTouchEvent, 0,0);
		return;
	}

	for(int i=0;i<juego.getSprites().size();i++){

		if(juego.getSprites().get(i).contains(pSceneTouchEvent.getX(),
		pSceneTouchEvent.getY())&&juego.getSprites().get(i).isVisible(
		)){
			if(msprt==null)	msprt=juego.getSprites().get(i);

	if(msprt.getZIndex()<juego.getSprites().get(i).getZIndex()){
				msprt=juego.getSprites().get(i);
				}

		}
	}
```

Clase Jugador

Es Una clase Abstracta, contiene los parámetros y métodos básicos que necesita cualquier jugador, de ella extienden los demás jugadores, JugadorHumano, JugadorBot y JuegadorOnline. A continuación podemos ver su constructor con los parámetros básicos que necesita, más adelante se le iniciara su baraja.

```
	public Jugador(Game game,int tipoJugador,int NumeroJugador, String
nombre ){
		/*
		 * tipoJugador 1 Humano
		 * tipoJugador 2 Bot
		 * tipoJugador 3 Remoto
		 */
		this.nombre=nombre;
		this.tipoJugador=tipoJugador;
		this.NumeroJugador=NumeroJugador;
		baraja =new Baraja();
		this.game=game;

	}
```

Clase JugadorHumano.

Es la clase que representa al usuario que está jugando, cuando es su turno espera a que el usuario realice una acción en la pantalla, que será capturada por el controlador táctil, y una vez finalizada la acción permitirá que la secuencia de la partida sigua su curso. En la siguiente figura podemos observar la función que se ejecuta cuando el jugador selecciona una carta, se encarga de llamar a la clase Director3Dpara que mueva las cartas por la pantalla, y se esta en modo Online se encarga de hacer una llamada a la clase DirectorOnline para que mande la acción a los jugadores que están jugando de forma remota.

```
publicvoid cartaSeleccionada(Carta carta){
      if(game.isOnline()){
            if(sacarCarta){


            game.getDirectorOnline().enviarCLI_PARTIDA_MANO_INI(this.getN
            ombre(), String.valueOf(carta.getNumero()),
            String.valueOf(carta.getSeleccion())) ;
            }else{
            game.getDirectorOnline().enviarCLI_PARTIDA_JUGADA(this.getNom
            bre(),String.valueOf(carta.getNumero()),String.valueOf(carta.
            getSeleccion())) ;
            }
      }
            cartaSalida=carta;
            cartaSalida.setSituacion(6);
            game.getDirector3D().CartaAretarHumano(carta);
            game.getDirectorJuego().getCD().countDown();
            sacarCarta=false;
            turno=false;
            Cartavista=false;
      }
```

Clase JugadorBot

Esta clase representa un jugador gobernado por la maquina, tiene un hilo de ejecución propio, cada jugadorBot tiene su propio hilo de ejecución con unos tiempos de espera de toma de decisiones aleatorio, de modo que cada jugador se tomara un tiempo distinto en elegir la carta que quiere seleccionar, así dará una impresión más humana al usuario.

Cuando el usuario empieza una ronda y saca una carta la case JugadorBot recibe la carta a la que tiene que hacer frente, internamente ordena en un array todas las cartas de las que dispone por el valor seleccionado y sitúa la primera que ganaría a la carta que selecciono el usuario como pivote, después dependiendo de la dificultad que tenga seleccionado este jugador (el valor de la dificultad se toma aleatoriamente cuando se inicie el jugador tendrá un nombre u otro en función de su dificultad asi los jugadores sabrán que jugadores son mas difíciles que otros), seleccionara un rango teniendo el pivote como centro y elegirá una carta aleatoriamente en ese rango, en la siguiente figura se muestra como seria gráficamente.

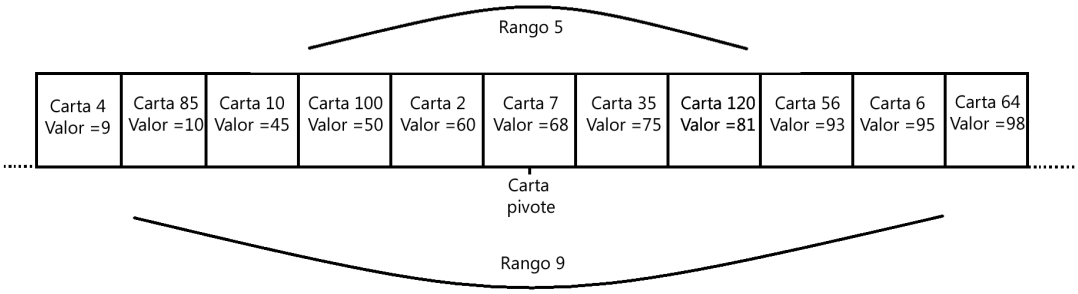


Ilustración 49: Mecanismo de elección de cartas.

Es este ejemplo grafico el usuario eligió una carta con un valor de 65, por lo que la primera carta del JugadorBot que gana a dicha carta es la carta 7 con un valor de 68, si la dificultad tuviera un rango 9, la carta que sacaría el JugadorBot seria elegida aleatoriamente entre ese rango, si el rango fuera 5, se elegiría una carta entre las del rango 5, mucho más ajustado.

Con el rango se trata de emular la posible memoria y conocimiento que un jugador podría tener sobre los países y sus valores.

Cuando el jugadorBot tiene que sacar una carta lo hace aleatoriamente, y para elegir el parámetro con el que jugara comparara la posición que tiene la carta según el parámetro seleccionara y para darle algo de aleatoriedad se utiliza el mismo rango porque si no siempreelegiría el mejor valor posible y jugaría como una máquina y no como una persona.

En la siguiente figura podemos ver una parte de la inteligencia Artificial.

```
publicvoid inteligencia2(){

	int numerocarta = tomaDecision(0, getNumeroCartas() - 1);
	cartaSalida= getBaraja().getList().get(numerocarta);
	MiSprite sprt=cartaSalida.getSprite();
	int valores[]=game.getBaraja().posiciones(cartaSalida.getSalud(),
	cartaSalida.getEducacion(), cartaSalida.getIngresos());
	int valor=cartaSalida.getPidh();
	sprt.setSeleccionBot(1);
	cartaSalida.setSeleccion(1);


	//unpocodealeatoridad
	valores[0]=tomaDecision(valores[0]-Rango, valores[0]+Rango);//
posicionsalud
	valores[1]=tomaDecision(valores[1]-Rango, valores[1]+Rango);//
posicioneducación
	valores[2]=tomaDecision(valores[2]-Rango, valores[2]+Rango);//
posicionidh

	if(valor>valores[0]){
		sprt.setSeleccionBot(2);
		cartaSalida.setSeleccion(2);
		valor=valores[0];
	}
	if(valor>valores[1]){
		sprt.setSeleccionBot(3);
		cartaSalida.setSeleccion(3);
		valor=valores[1];
	}
	if(valor>valores[2]){
		sprt.setSeleccionBot(4);
		cartaSalida.setSeleccion(4);
		valor=valores[2];
	}
}
```

Clase JugadorRemoto

Este jugador espera a que le llegue un mensaje de la clase directorOnline para saber que acción ha realizado el jugador que está jugando de forma remota y ejecuta la misma acción. Aquí podemos ver una clase interna que se encarga de esperar a que el jugador remoto envié el mensaje de la carta ha sacado para hacerlo él.

```
publicclassAccionSincro {

	privatebooleanaccion=false;
	privateintcarta=-1;
	privateintseleccion=-1;


	publicsynchronizedint[] esperaAccion() {
		try {
			while (!accion) {
				this.wait();
			}
		} catch (InterruptedException e) {

			e.printStackTrace();
		}
		accion=false;
		int temp[]=newint[2];
		temp[0]=carta;
		temp[1]=seleccion;
		carta=-1;
		seleccion=-1;
		return temp;

	}
```

Clase ClienteTCP

La clase clienteTCP es la encargada de gestionar la conexión con el servidor. Se encarga de escuchar los mensajes que le llegan así como de enviar los mensajes que se necesiten enviar, en la siguiente figura podemos ver parte del código que envía un mensaje al servidor.

```
publicvoid enviarCLI_MODIFICAR(final String... parametros) {
	System.out.println("enviarCLI_MODIFICAR");
	Thread hilo =new Thread (){
	publicvoid run(){
		sincro.EsperaIniciado();
		StringBuffer output = new StringBuffer();
output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
output.append("<mensaje protocol_ver=\"1.0\">");
output.append("<mensaje_ID>CLI_MODIFICAR</mensaje_ID>");
output.append("<datos>");
output.append("<usuario>" + parametros[0]+ "</usuario>");
output.append("<password>" + parametros[1]+ "</password>");
output.append("<new_password>" + parametros[2]+ "</new_password>");
output.append("<nombre>" + parametros[3]+ "</nombre>");
output.append("<apellidos>" + parametros[4]+ "</apellidos>");
output.append("<pais>" + parametros[5]+ "</pais>");
output.append("</datos>");
output.append("</mensaje>");
enviarDatos(output.toString());

...
```

La siguiente función muestra los pasos que sigue para iniciar una conexión con el servidor.

```
	publicvoid ejecutarCliente() {
		desconexion=false;
		System.out.println("\nejecutarCliente()");
		// conectarse al servidor, obtenerflujos, procesar la conexión
		try {
		conectarAServidor();//Paso 1: crea run socket
		obtenerFlujos();// Paso 2: obtener los flujos
		procesarConexion();// Paso 3: procesar la conexión
		}
		catch (EOFException excepcionEOF) {
			System.err.println("El cliente termino la conexión");
		}
		catch (IOException excepcionES) {
			excepcionES.printStackTrace();
		} finally {
			cerrarConexion(); // Paso 4: cerrar la conexión
		}
	} // fin del método ejecutarCliente
```

Clase ParserTCP

Es la clase encargada de parsear los mensajes que llegan a la aplicación.

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                    builder = factory.newDocumentBuilder();
                    Document dom = builder.parse(new InputSource(new
StringReader(mensaje)));

                    Element root = dom.getDocumentElement();
                    NodeList mensajes = root.getElementsByTagName("mensaje");
                    Node nodo = mensajes.item(0);

                    Document documento = builder.parse(new InputSource(new
StringReader(mensaje)));

      NodeList listaNodos = documento.getElementsByTagName("mensaje_ID");
            for (int i = 0; i < listaNodos.getLength(); i++) {
      tipo_mensaje=listaNodos.item(i).getTextContent();
```

Lo primero que hace es averiguar de qué tipo de mensaje es y luego llama a una función que procesa el tipo de mensaje. A continuación vemos el código del procesamiento de datos del mensaje de inicio de ronda.

```
publicvoid SERV_PARTIDA_MANO_INI(){

      for (int j = 0; j <parametros.getLength(); j++) {
            if(parametros.item(j).getNodeName().trim().equals("turno")){
                  datos[0]=parametros.item(j).getTextContent();
            }
            if(parametros.item(j).getNodeName().trim().equals("carta")){
                  datos[1]=parametros.item(j).getTextContent();
            }
      if(parametros.item(j).getNodeName().trim().equals("seleccion")){
                  datos[2]=parametros.item(j).getTextContent();
            }
      }

}
```

Clase Camera3D.

Esta clase extiende de la clase Camera, lo que hace es reconfigurar la perspectiva para que tenga una perspectiva cónica mediante esta función.

```
privatevoid setFrustum(GL10 pGL)
	{

	// setea el campo de vision a 60 grados
	float fov_grados = 60;
	float fov_radianes = fov_grados / 180 * (float)Math.PI;

	// configura el aspecto y la distancia de la camara
	float aspecto = this.getWidth() / this.getHeight();
	float camZ = this.getHeight()/2 / (float)Math.tan(fov_radianes/2);

	// configura la  proyeccion
		GLHelper.setProjectionIdentityMatrix(pGL);
	    GLU.gluPerspective(pGL, fov_grados, aspecto, camZ/10, camZ*10);

	// configura la vista
		GLU.gluLookAt(pGL 0,  0, camZ, 0, 0,  0,  0,   1, 0);

		pGL.glScalef(1,-1,1);
	pGL.glTranslatef(-800/2,-480/2,0); // pone el origen de coordenadas

	}
```

Clase Baraja:

La clase Baraja contiene todas las cartas, inicialmente se crea una baraja con todas las cartas, mas tarde se van sacando las cartas de la baraja inicial y se las añade de la baraja de cada jugador. Las cartas son guardadas en un ArrayList como puede verse en el constructor.

```
	public Baraja(){
		list= new ArrayList<Carta>();
		generator = new Random();
	}
```

Contiene métodos útiles para la cuantificación de los valores de las cartas, por ejemplo esta función va guardando la media acumulativa de las cartas que se van añadiendo a la baraja.

```
	publicboolean add(Carta object) {
		mediaEDu+=object.getEducacion();
		mediaidh+=object.getIdh();
		mediapib+=object.getIngresos();
		mediasalud+=object.getSalud();
		paises++;
		returnlist.add(object);
	}
```

Con esta función a unos valores dados de salud, PIB y educación nos devolverá la posición global que tiene los valores de la carta, con este modo el JugadorBot podrá elegir el mejor parámetro de la carta a elegir.

```
publicint[] posiciones(Float salud, Float edu, Float pib){
	int possalud=1;
	int posedu=1;
	int pospib=1;
	for ( Iterator<Carta> it = list.iterator();it.hasNext();){
		Carta carta =(Carta)it.next();
		if(salud<carta.getSalud()) possalud++;
		if(edu<carta.getEducacion()) posedu++;
		if(pib<carta.getIngresos()) pospib++;
	}
	int salida[]={possalud,posedu,pospib};
	return salida;
}
```

El siguiente código reparte las cartas entre los jugadores.

```
publicvoid reparteVerCartas(){
	HashSet<String> repartida=new HashSet<String>();
	ArrayList<Carta> listaTemp = new ArrayList<Carta>();
	intmax=list.size();
	boolean metido;

	for(int i=0; i<187;i++){

			int index=-1;
			do{
				index=186-i;

	metido=repartida.add(list.get(index).getNombre());
			}while(!metido);
			list.get(index).getSprite().setVisible(false);
			list.get(index).setJugador(1);
	list.get(index).getSprite().registrar()
	listaTemp.add(list.get(index));

	}
	list = new ArrayList<Carta>();
	list=listaTemp;

	//comprobacion
	System.out.println("la lista tiene "+list.size());

	for ( Iterator<Carta> it = list.iterator();it.hasNext();){
		Carta carta =(Carta)it.next();
System.out.println("el jugador " +carta.getNumJugador()+" tiene " +
carta.getNombre());

	}
}
```

La case MiSprite:

Esta es la clasemás importante del apartado grafico, es la clase que modela las cartas.

Contiene las texturas de las banderas, el fondo de las cartas, el reverso y los dados de las cartas, cada de estas entidades cuenta con su propia área táctil.

Es muy importante la coordinación de todas sus texturas internas ya que de lo contrario, cuando se moviera la carta veríamos como sus texturas internas actualizarían su posición a destiempo y veríamos como si los datos o la bandera no están sincronizadas, lo que sería muy molesto. La función que se encarga de sincronizar la actualización de todas las texturas es la siguiente.

```
Protected void onManagedUpdate(float pSecondsElapsed) {
//para optimizar el rendimiento se hacen aparecer y desaparecer las cartas
del usuario humano
      if(carta.getSituacion()==3){
            if(getBocaAbajo()&&getRecienPuestoBocaAbajo()){

            int pos
      =juego.getDirectorJuego().gethumano().getBaraja().getPosicion(carta);
                  try{

      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos+3)
.getSprite().setVisible(false);
                  }catch(Exception e){}

                  try{
            juego.getDirectorJuego().gethumano().getBaraja().getList().get
      (pos-1).getSprite().setVisible(true);
      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
      2).getSprite().setVisible(true);
      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
      3).getSprite().setVisible(true);
                  }catch(Exception e){}
                        setRecienPuestoBocaAbajo(false);
                  }

                  if(!getBocaAbajo()&&getRecienPuestoBocaArriba()){

                  int pos
                  =juego.getDirectorJuego().gethumano().getBaraja().getPos
                  icion(carta);
                  try{
                        juego.getDirectorJuego().gethumano().getBaraja().
                  getList().get(pos+1).getSprite().setVisible(true);
                        juego.getDirectorJuego().gethumano().getBaraja().
                  getList().get(pos+2).getSprite().setVisible(true);
                        juego.getDirectorJuego().gethumano().getBaraja().
                  getList().get(pos+3).getSprite().setVisible(true);
                        }catch(Exception e){}

                  try{

                  juego.getDirectorJuego().gethumano().getBaraja().getList
            ().get(pos-3).getSprite().setVisible(false);
                  }catch(Exception e){}
                        setRecienPuestoBocaArriba(false);
                  }
                  mirarcartas();
                  sincronizaTexturas();
            }
super.onManagedUpdate(pSecondsElapsed);

      }
```

Cada región táctil que contiene la carta se gestiona con una función que tiene consciencia de la escena y el momento en que se encuentra, por ejemplo no es lo mismo tocar la carta un jugador ha empezado la mano y ha seleccionado una carta con un valor que cuando es el usuario el que tiene que sacar una carta y elegir el valor con el que se va a jugar, por ese motivo la carta tiene que ser consciente de que acciones se le pueden pedir. A continuación se muestra un fragmento de la función ya que la función entera tiene mas de 200 líneas de código.

```
Public Boolean onAreaTouched(final TouchEvent pSceneTouchEvent, finalfloat
pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
	if (!juego.getDirectorTactil().pillar(numero)) {
		returnfalse;
	}
//para que no se puedan tocar otras cartas si seleccionar una carta para jugar
	if(juego.getDirectorJuego().gethumano().CartaVista()&&(carta.getSituacio
	n()!=4&&carta.getSituacion()!=5)){
			returnfalse;
	}
	if (carta.getSituacion()==0)returnfalse;

	switch (pSceneTouchEvent.getAction()) {

	case 0: // registra la posicion inicial de la carta y el tiempo
		oldX = (int) pSceneTouchEvent.getX();
		oldY = (int) pSceneTouchEvent.getY();
		(…)
		break;
	case 1:
		(...)//funciones que recolocaran la carta cuando seasoltada
			// si la carta tocada no es la del usuario
		}
		if (carta.getSituacion() ==
		3||carta.getSituacion()==4||carta.getSituacion()==5) {
		// cuando sueltas mira el angulo y deja las cartas en su
		// posicion
			long tiempoArrastre = System.currentTimeMillis() - tiempo;
			long distancia = 3 * Math.abs(oldX - (int)
pSceneTouchEvent.getX());
			long velocidad = distancia / tiempoArrastre;
			if (velocidad >= 1) movidoRapido = true;
		// SI LANZAS LA CARTA LLEGARA A SU SITIO SOLA
		(...) // codigo que recoloca la carta si esta se solto con inercia
			//la ponemos para coger su seleccion
		if (click()&&!getBocaAbajo()&&carta.getSituacion() ==
	3&&!carta.getJugador().CartaVista()){
			scene.registerTouchArea(PIDH);
			scene.registerTouchArea(salud);
			scene.registerTouchArea(educacion);

	scene.registerTouchArea(pib);if(carta.getJugador().getTipojugador()==1&&
carta.getJugador().turno()){

		}
	// PARTE QUE LE PERMITE SOLTAR LA CARTA Y QUE SE COLOQUE SOLA EN LA
	// POSICION SI NO SE SOLTO CON INERCIA
			if (movido && !movidoRapido&&carta.getSituacion() == 3) {
				if ((getRotation() <= -90f)) {
					registerEntityModifier(new MoverX(0.4f,
getX(), 91) {
						protectedvoid onModifierFinished(IEntity
pItem) {
							MiSprite df = (MiSprite) pItem;
							df.setRecienPuestoBocaAbajo(true);
							super.onModifierFinished(pItem);

(…)
}
```

5.4 PRUEBAS Y DEPURACIÓN DE LA APLICACIÓN

El proceso de realización de pruebas en un principio se hizo en su fase inicial con el emulador que viene de serie con el Android SDK. No obstante, la lenta ejecución del sistema operativo hacía imposible saber el rendimiento real de la aplicación en fotogramas por segundo (FPS). Por ello, después se utilizó un emulador de Android de un sistema operativo Froyo 2.2 compilado para x86 corriendo en Virtualbox. Sin conseguir resultados satisfactorios en cuanto rendimiento, debido a que la propia ejecución del sistema operativo en el emulador aunque mejor que en emulador de serie del Android SDK seguía siendo lenta, más tarde se integró en dos Smartphones para realizar las pruebas y depuración, un Galaxy Mini y un HTC Wildfire S de muy diferentes prestaciones. Se obtuvieron mejores resultados de rendimiento en dichos dispositivos con tasas estables entrono a 10-15 FPS.Como el objetivo de este PFC era llegar al máximo público posible se realizaron los cambios en la gestión de texturas y de Sprites y de las superficies táctiles que hicieran compatible al app con diferentes modelos y terminales de gama baja.

Dado que se necesitaba tener creadas 187 cartas con sus datos y sus texturas para el desarrollo de la partida aunque solo son visibles una pocas a la vez. En los mazos de las cartas de los usuarios existe mucho solape, ósea que hay muchas cartas una encima de otra. Se decidió utilizar un técnica que descargaba de memoria gráfica todas las texturas que estuvieran por debajo de dos cartas, mientras se seguían encontrando en memoria principal, mucho más abundante y así evitar tener que recargar las texturas desde el disco , lo que llevaría a que la aplicación diera tirones al pasar las cartas.

Esta técnica consiste en que según se van pasando las cartas de un lado a otro detecta a qué lado se están pasando las cartas y cuando hay 3 apiladas aplica un detach a la tercera por arriba, por el otro lado de donde se van sacando las cartas si se detecta que solo quedan dos cartas visibles y se realiza un attach de una tercera. Con esta técnica se incremento el rendimiento grafico hasta los 30-40 FPS.

Algo parecido ocurría con las superficies táctiles, ya que se cuenta con 187 cartas, cada carta con 5 superficies táctiles independientes, cuatro para los datos y una global para la carta, lo que nos da 935 superficies táctiles solamente de las cartas, se observó que la respuesta táctil tenía un pequeño Lag, el tiempo entre que se toca la carta y el sistema se da cuenta de que se ha tocado dicha carta esa de unas decimas de segundo. Por ello se decidió modificar el controlador táctil para que utilizara la misma técnica que en las texturas dando como resultado una magnifica respuesta táctil y una mayor tasa de refresco en torno a 40-50 FPS. Teniendo en cuenta que estos dispositivos son de gama baja se considero que la aplicación ya tenía un rendimiento excelente incluso en dispositivos viejos y poco potentes.

Para las pruebas con el servidor se ejecutó una aplicación de servidor en un servidor de la escuela y se realizaron partidas, en todo momento se mostraba por pantalla el intercambio de mensajes, el uso de XML para los mensajes simplifico mucho la tarea de depuración ya que se podía leer fácilmente el flujo de mensajes.

La realización de la aplicación se ha realizado de forma incremental, asegurando que el trabajo hecho funcionara correctamente y teniendo un repositorio de versiones. Las pruebas que se han realizado han cubierto tanto la correcta iniciación de la aplicación en diferentes terminales, la correcta carga de texturas y de los archivos de control, el correcto registro y verificación de datos del usuario, numerosas partidas mono jugador y online a fin de asegurar un correcto intercambio de mensajes para la realización de partidas sin incidentes, en el apartado grafico se ha verificado en diferentes terminales una correcta visualización, Pruebas de rendimiento y tiempos de carga de datos.

En el proceso de depuración y pruebas se realizó con varios dispositivos, todos ellos con diferentes resoluciones y proporciones de pantalla, lo que fue muy útil para asegurar un correcto funcionamiento y visualización del programa en los diversos dispositivos que hay en el mercado.

La participación de un grupo de 10 usuarios, 7 hombres y 3 mujeres, en el proceso de validación final ha permitido depurar no sólo el rendimiento de la aplicación sino también su funcionamiento validando así las expectativas de usabilidad. Todos los usuarios fueron capaces de usar la aplicación por sí solos sin necesidad de preguntas. La implementación de un tutorial permitió resolver sus dudas de forma autónoma.

A continuación se muestran los tres dispositivos más representativos validados ysus principales características:

Samsung Galaxy Mini S5570

Características:

DISPLAY: 240 x 320 pixels, 3.14 pulgadas, Capacitiva
PROCESADOR: Qualcomm MSM7227 ARMv6 600MHz,
GPU: Adreno 200
RAM: 384MB
SO: Android 2.2 Froyo

HTC Wildfire S

Características:

DISPLAY: 320 × 480pixels, 3.2 pulgadas, Capacitiva

PROCESADOR: ARMv11 a600MHz,

GPU: Adreno 200

RAM: 512 MB

SO: Android 2.3 Gingerbread

Samsung Galaxy S3 i9300

Características:

DISPLAY: 720 × 1280 pixels, 4.7 pulgadas, Capacitiva

PROCESADOR: Exynos 4212 1.4GHz (QuadCore)

GPU: Mali-400MP

RAM: 1024 MB

SO: Android 4.1.2 Jelly Bean

6 CONCLUSIONES Y FUTURAS MEJORAS

6.1 CONCLUSIONES

La realización de este PFCha permitido profundizar en el conocimiento de los problemas de diseño, implementación y validación de un app en entornos móviles reales con Android para una aplicación de algo más de 8000 líneas de código.

Se ha verificado la efectividad y eficiencia que tiene una adecuada planificación y diseño desde el inicio, aunque no siempre ha sido posible llevar el diseño a la implementación debido al rediseño de algunas partes por carencias y errores que ha mostrado AndEngine y el hándicap que supone la falta de documentación que fuerza al aprendizaje sobre la marcha de este framework según las necesidades en ese momento.

Durante la realización de este PFC he profundizado en tecnologías clave como XML, 3D y Modelo cliente-servidor constando la utilidad de dichas facilidades para vídejeugos en móvil con arquitecturas cliente-servidor.

El diseño y desarrollo del protocolo XML ha servido para profundizar en el funcionamiento y colaboración dinámica entre un cliente y un servidor resaltando la necesidad de separación de tareas que tiene que tener un juego multijugador con su servidor así como l tratamiento que tienen que hacerse a los flujos que comunican ambas aplicaciones, además de haber integrado en el tratamiento de archivos XML.

La creación de un videojuego es una tarea ardua y multidisciplinar, una persona que desarrolle un videojuego debe saber diseñar y codificar una aplicación pero hay muchas otras tareas en el desarrollo de un videojuego. Se han tenido que diseñar todos los gráficos del juego mediante programas de edicióngráfica y se ha tenido que aprender a realizar las animaciones que tiene el juego y validar su efectividad con usarios reales.

Durante el desarrollo de esta aplicación he tenido que jugar y observar muchas partidas para verificar que todo funcionara correctamente. En este periodo ha aumentado mi interés por el IDH, y desde el punto de vista de usuario se valida que es una herramienta efectiva para la concienciación de las personas sobre el desarrollo humano, misión esencial de este PFC.

Android.

La elección de un lenguaje conocido como java para la programación de aplicaciones facilita mucho que múltiples programadores se decidan por Android para hacer sus aplicaciones y formen una gran comunidad. No obstante la forma en que se programaría un videojuego directamente en Android llevaría mucho tiempo debido a la falta de funciones específicas para juegos.

Los juegos realizados de esta forma no tendrían el mismo rendimiento que usando un framework para Android. Las últimas versiones de Android 4.X disponen de aceleración hardware, pero hoy en día hay muchos dispositivos con versiones inferiores que no aprovechan esa ventaja. Google debería ofrecer un framework propio que ofreciera las ventajas de los frameworks y combinarlo con documentación suficiente.

AndEngine.

Ofrece muchas opciones a la hora de diseñar un videojuego a parte de la aceleración hardware. Aunque está diseñado para funcionar en 2D puede ser modificado para trabajar en 3D, lo que puede ser muy interesante de cara a hacer efectos 3D en juegos 2D, que no podrían realizarse directamente en Android. Presenta muchas carencias si pretende ser aceptado por la comunidad.

La carencia de Documentación hace tremendamente difícil empezar a utilizarlo, y es una queja omnipresente en cualquier foro sobre este framework, la versión utilizada fue la versión Open GL ES 1.1 cuyo desarrollo ha sido detenido, la versión actualmente en desarrollo la Open GL ES 2 tiene aún muchos fallos de incompatibilidad lo que no la convierten en la opción elegida si se pretende crear una aplicación que pueda llegar a la mayor cantidad de público posible. En mi opinión tiene aun mucho camino por andar y deben solucionar el problema de la documentación.

6.2 FUTURAS MEJORAS

Tras la realización de este PFC, se han recogido diferentes líneas de trabajo que podrían desarrollarse en un futuro, de tipo técnico y funcional.

I) Mejoras Funcionales

Creación de un CHAT: El protocolo y el cliente están preparados para esta funcionalidad, pero debido a problemas con el framework AndEngine ya que no existe una clase equivalente a un TextBox, se trato de construir una case propia que realizara tal función pero daba problemas de representación en pantallas de diferentes tamaños y se dejo para una futura mejora.

Integración con Redes Sociales: La aplicación podría estar conectada a una cuenta del usuario de una red social, para por ejemplo poder jugar con las personas que tenga esa persona en dicha red social.

Acceso al foro desde la aplicación: De este modo las personas podrían estar en un canal del foro del servidor y charlar entre ellos, cuando dos usuarios deseasen jugar podrían crear una partida he invitar al resto de los jugadores que deseen.

Multijugador Bluetooth: Añadir una nueva forma de juego en el que se utilice Bluetooth para comunicar a los dispositivos.

Elección de rondas: Hacer que el numero de rondas se pueda fijar al inicio de la partida para jugar partidas rápidas o completas.

II) Mejoras técnicas

Optimización para Tablet: Dado que las tablets tienen un tamaño de pantalla mucho más grande, aunque sean compatibles con la versión actual del juego, el tamaño de las cartas cuando están desplegadas podría resultar muy grandes, una mejora seria que la aplicación detectara si es una tablet y re calculará el tamaño que deberían tener las cartas.

Traducción de Textos: Actualmente la aplicación se encuentra en castellano. Se preveía su traducción al inglés, sin embargo por falta de tiempo se dejó como una futura mejora.

7 BIBLIOGRAFIA

1. Programa de las Naciones Unidas para el Desarrollo, “El PNUD en breve”, 2012.
   Disponible en: http://www.undp.org/content/dam/undp/library/corporate/brochure/w-undp_brochure_2012-Spanish-final.pdf

2. Naciones Unidas, Objetivos del milenio, 2013.
   Disponible en: http://www.un.org/es/millenniumgoals/

3. Naciones Unidas, Preguntas frecuentes sobre el IDH, 2013.
   Disponible en: http://hdr.undp.org/es/estadisticas/faq/

4. Naciones Unidas, Estadísticas sobre el IDH, 2013.
   Disponible en: http://hdr.undp.org/es/estadisticas/idh/

5. Programa de las Naciones Unidas para el Desarrollo, Informe del IDH de 2013.
   Disponible en : http://hdr.undp.org/es/

6. Asociación Española de Distribuidores y Editores de Software de Entretenimiento , Estudio sobre los videojuegos, 2012.
   Disponible en: http://www.adese.es/docs/documentacion/estudios-y-analisis

7. Página oficial del proyecto Phonegap.
   Disponible en: http://www.phonegap.com

8. Gertner, Estudio sobre la penetrecion de Android, 2013.
   Disponible en : http://www.gartner.com/newsroom/id/2120015

9. Miguel Ángel Álvarez, “Introducción a XML”, 2011.
   Disponible en : http://www.desarrolloweb.com/manuales/18/

10. Foro oficial de AndEngine
    Disponible: http://www.andengine.org/forums/

11. Página de la wikipedia sobre XML, 2013.
Disponible en: http://es.wikipedia.org/wiki/Extensible_Markup_Language

12. ITU, Informe sobre el estado de las telecomunicaciones 2013.
Disponible en: http://www.itu.int/net/pressoffice/press_releases/2013/05-es.aspx#.UiceqDbwnuX

13. Alberto Santos Estevez, " tipos de aplicaciones móviles", 2011.
Disponible en : http://geospatialtrainginges.com/recursos-gratuitos/tipos-de-aplicaciones-moviles/

14. Página principal de la Open Handset Alliance
Disponible en: http://www.openhandsetalliance.com/

15. Wikipedia, Historia e información de Android, 2013.
Disponible en: http://es.wikipedia.org/wiki/Android

16. Jairo Chapela Martínez, Manual de uso e información sobre Eclipse, 2007.
Disponible en: http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf

17. ONG, Educación para el desarrollo
Disponible en: http://www.educacionparaeldesarrollo.org/

18. ONG educación para el desarrollo, Juegos cooperativos
Disponible: http://www.enredate.org/

19. Notepad++
Disponible en: http://notepad-plus-plus.org/

20. Roberto Albornoz Figueroa, "Conceptos básicos para desarrollo de videojuegos en 2D", 2007.
Disponible en : http://www.etnassoft.com/biblioteca/desarrollo-de-videojuegos/

21. José Antonio Roig Torres, "Diseño e implementación de un sistema de intercambio electrónico de datos (EDI) sobre redes TCP/IP", 2003.
Disponible en : http://www.interibiza.com/tecno/pfc/index.htm

ANEXO A. Listado de acrónimos

W3C: World Wide Web Consortium

DTD: Definición de tipo de documento

ACK: Acknowledgement

SDK: Software development kit

SGML: Standard Generalized Markup Language

XML: Extensible Markup Language

UML: Unified Modeling Language

CSS: Cascading Style Sheet

PFC: Proyecto Fin Carrera

IDH: Índice de Desarrollo Humano

HTML: HyperText Markup Language

ANEXO B. Listado del código fuente

Archivo Game.java

```
package xnetcom.pro.cartas.activities;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.Semaphore;

import javax.microedition.khronos.opengles.GL10;
import org.anddev.andengine.audio.sound.Sound;
import org.anddev.andengine.audio.sound.SoundFactory;
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.engine.options.EngineOptions;
import org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import
org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy;
import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.EntityBackground;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.text.ChangeableText;
import org.anddev.andengine.input.touch.TouchEvent;
import org.anddev.andengine.opengl.font.Font;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlas;
import
org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlasTextureReg
ionFactory;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.opengl.util.GLHelper;
import org.anddev.andengine.ui.activity.BaseGameActivity;
import org.anddev.andengine.util.Debug;
```

```
import xnetcom.pro.cartas.R;
import xnetcom.pro.cartas.auxiliares.CamaraEscena;
import xnetcom.pro.cartas.auxiliares.Camera3d;
import xnetcom.pro.cartas.auxiliares.Parser_Baraja;
import xnetcom.pro.cartas.auxiliares.datosUsuario;
import xnetcom.pro.cartas.juego.Baraja;
import xnetcom.pro.cartas.juego.Carta;
import xnetcom.pro.cartas.juego.Director3D;
import xnetcom.pro.cartas.juego.DirectorJuego;
import xnetcom.pro.cartas.juego.DirectorTactil;
import xnetcom.pro.cartas.red.DirectorOnline;
import xnetcom.pro.cartas.red.HiloTCP;
import xnetcom.pro.cartas.sprites.BlogSprite;
import xnetcom.pro.cartas.sprites.Dado;
import xnetcom.pro.cartas.sprites.Delay;
import xnetcom.pro.cartas.sprites.MiSprite;
import xnetcom.pro.cartas.sprites.TicSprite;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.InputMethodManager;
import android.widget.Toast;

publicclass Game extends BaseGameActivity {

	// ===========================================================
	// constantes
	// ===========================================================

	publicstaticfinalintCAMERA_WIDTH = 800;
	publicstaticfinalintCAMERA_HEIGHT = 480;

	// ===========================================================
	// manejadores
	// ===========================================================
	private DirectorTactil tactil;
	private Director3D director3d;
	private DirectorJuego directorjuego;

	// ===========================================================
	// escenas
	// ===========================================================

	private Camera3d CameraGame;
	private Camera Camera;
	private CamaraEscena escenaJuego;

	// ===========================================================
	// Texturas
```

```
// ============================================================
private BitmapTextureAtlas[] BTAflags;
private BitmapTextureAtlas BTAFondoJuego;
private BitmapTextureAtlas BTAChat;
private BitmapTextureAtlas BTAblog;
private BitmapTextureAtlas BTAdado;
private BitmapTextureAtlas BTAtic;
private BitmapTextureAtlas BTAcarta;
private BitmapTextureAtlas BTAdorso;
private BitmapTextureAtlas BTAtimer;
private BitmapTextureAtlas mFontTexture;
private BitmapTextureAtlas mFontTextureGrande;
private BitmapTextureAtlas mFontTextureBlue;
private BitmapTextureAtlas mFontTexture2;
private BitmapTextureAtlas mFontTextureChat;

// ============================================================
// regionesdetextura
// ============================================================
private TextureRegion[] TRpaises;
private TextureRegion TRFondoJuego;
private TextureRegion TRblog;
private TextureRegion TRChat;
private TextureRegion TRChat_superior;
private TextureRegion TRChat_back;
private TextureRegion TRcarta;
private TextureRegion TRdorso;
private TextureRegion TRportada;

private TiledTextureRegion TTRdado;
private TiledTextureRegion TTRtic;
private TiledTextureRegion TTRtimer;

// ============================================================
// sonidos
// ============================================================
private Sound Sonido_Dados;

// ============================================================
// sprites
// ============================================================

private Sprite SPRchat;
private AnimatedSprite timer;
private TicSprite tic;
private Dado dado;
private BlogSprite blog;
private ArrayList<MiSprite>listSprite = new ArrayList();

// ============================================================
// fuentes
// ============================================================
public Font mFont;
public Font mFontBlue;
public Font mFont2;
public Font mFontChat;
public Font mFontGrande;

// ============================================================
```

```
// textos
// ============================================================
private ChangeableText chat;

// ============================================================
// barajacontodaslascartas
// ============================================================
private Baraja baraja;

// ============================================================
// otros
// ============================================================

private datosUsuario datos;
private HiloTCP hilotcp;
privatebooleanonline = false;
private DirectorOnline DO;



// ============================================================
// metodos
// ============================================================
publicboolean isOnline() {
	returnonline;
}

publicvoid setDirectorOnline(DirectorOnline DO) {
	this.DO = DO;
}

public DirectorOnline getDirectorOnline() {
	returnDO;
}

public BlogSprite getblog() {
	returnblog;
}

public Baraja getBaraja() {
	returnbaraja;
}

public Font getFontBlue() {
	returnmFontBlue;
}

public Font getFont() {
	returnmFont;
}

public Font getFontGrande() {
	returnmFontGrande;
}

public CamaraEscena getScene() {
	returnescenaJuego;
}
```

```
	public DirectorTactil getDirectorTactil() {
		returntactil;
	}

	public Director3D getDirector3D() {
		returndirector3d;
	}

	public TextureRegion getDorso() {
		returnTRdorso;
	}

	public DirectorJuego getDirectorJuego() {
		returndirectorjuego;
	}


	public ArrayList<MiSprite> getSprites() {
		returnlistSprite;
	}

	public TicSprite getTic() {
		returntic;
	}

	@Override
	public Engine onLoadEngine() {

		this.Camera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);

		returnnew Engine(
				new EngineOptions(true,
ScreenOrientation.LANDSCAPE,
					new
RatioResolutionPolicy(CAMERA_WIDTH, CAMERA_HEIGHT),
					this.Camera).setNeedsSound(true));

	}

	@Override
	publicvoid onLoadResources() {
		// TODO Auto-generated method stub
		this.BTAtimer = new BitmapTextureAtlas(2048, 256,
				TextureOptions.NEAREST);

		BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");
		this.TTRtimer = BitmapTextureAtlasTextureRegionFactory
				.createTiledFromAsset(this.BTAtimer, this,
"timer.png", 0, 0,
					12, 1);
		this.mEngine.getTextureManager().loadTexture(this.BTAtimer);
		CargaRecursos();
	}

	privatevoid CargaRecursos() {

		this.BTAblog = new BitmapTextureAtlas(1024, 512,
				TextureOptions.BILINEAR_PREMULTIPLYALPHA);
```

```
            this.BTAtic = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            this.BTAChat = new BitmapTextureAtlas(1024, 1024,
                        TextureOptions.NEAREST);
            this.BTAdado = new BitmapTextureAtlas(128, 32,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.BTAFondoJuego = new BitmapTextureAtlas(1024, 512,
                        TextureOptions.NEAREST);
            this.BTAcarta = new BitmapTextureAtlas(512, 512,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.BTAdorso = new BitmapTextureAtlas(512, 512,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);


      BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/botones/"
);

            BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");
            this.TRblog =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(
                        this.BTAblog, this, "blog_4.png", 0, 0);
            this.TRcarta =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(
                        this.BTAcarta, this, "carta_v4.png", 0, 0);
            this.TRdorso =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(
                        this.BTAdorso, this, "dorso3.png", 0, 0);
            this.TRFondoJuego = BitmapTextureAtlasTextureRegionFactory
                        .createFromAsset(this.BTAFondoJuego, this,
                                    "tapete_tablon3.png", 0, 0);
            this.TTRdado = BitmapTextureAtlasTextureRegionFactory
                        .createTiledFromAsset(this.BTAdado, this,
"dado.png", 0, 0, 4,
                                    1);
            this.TTRtic = BitmapTextureAtlasTextureRegionFactory
                        .createTiledFromAsset(this.BTAtic, this,
"tics.png", 0, 0, 2, 1);
            this.TRChat =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(
                        this.BTAChat, this, "chat_blanco.png", 0, 0);
            this.TRChat_superior = BitmapTextureAtlasTextureRegionFactory
                        .createFromAsset(this.BTAChat, this,
"chat_superior.png", 0,
                                    500);
            this.TRChat_back = BitmapTextureAtlasTextureRegionFactory
                        .createFromAsset(this.BTAChat, this,
"chat_back.png", 0, 600);

            director3d = new Director3D(this);
            InputStream fraw = getResources().openRawResource(R.raw.baraja);

            Parser_Baraja par = new Parser_Baraja();
            baraja = par.parse(fraw);

            System.out.println("baraja.getList().size()" +
baraja.getList().size());

            int numero = (baraja.getList().size() / 8) + 1;
```

```
            System.out.println("numero" + numero);

            BTAflags = new BitmapTextureAtlas[numero];

            for (int i = 0; i < numero; i++) {
                  this.BTAflags[i] = new BitmapTextureAtlas(256, 1024,
                              TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            }

            this.TRpaises = new TextureRegion[baraja.getList().size()];


      BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/flags/");

            int i = 0;
            int j = 0;
            int h = 0;
            for (Iterator<Carta> it = baraja.getList().iterator();
it.hasNext(); i++, h++) {
                  Carta carta = (Carta) it.next();
                  if (i == 8) {
                        i = 0;
                        j++;
                  }
                  TRpaises[h] = BitmapTextureAtlasTextureRegionFactory
                              .createFromAsset(this.BTAflags[j], this,
carta.getArchivo()
                                          + ".png", 0, i * 128);

                  System.out.println("montando en la posicion " + (i * 128)
                              + "el pais siguente");
                  Log.i("cargando", carta.getArchivo() + ".png");
            }

            SoundFactory.setAssetBasePath("mfx/");
            try {
                  this.Sonido_Dados = SoundFactory.createSoundFromAsset(
                              this.mEngine.getSoundManager(), this,
"dados.mp3");

            } catch (final IOException e) {
                  Debug.e(e);
            }

            this.mFontTexture = new BitmapTextureAtlas(1024, 256,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.mFontTextureGrande = new BitmapTextureAtlas(512, 256,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.mFontTextureBlue = new BitmapTextureAtlas(1024, 256,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.mFontTexture2 = new BitmapTextureAtlas(1024, 256,
                        TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.mFont = new Font(this.mFontTexture, Typeface.create(
                        Typeface.DEFAULT, Typeface.BOLD), 24, true,
Color.BLACK);
            this.mFontGrande = new Font(this.mFontTextureGrande,
Typeface.create(
                        Typeface.DEFAULT, Typeface.NORMAL), 48, true,
Color.BLACK);
```

```
            this.mFontBlue = new Font(this.mFontTextureBlue,
Typeface.create(
                        Typeface.DEFAULT, Typeface.BOLD), 24, true,
Color.BLUE);
            this.mFont2 = new Font(this.mFontTexture2, Typeface.create(
                        Typeface.DEFAULT, Typeface.BOLD), 32, true,
Color.BLACK);

            this.mFontTextureChat = new BitmapTextureAtlas(1024,
1024,TextureOptions.BILINEAR_PREMULTIPLYALPHA);
            this.mFontChat = new Font(this.mFontTextureChat,
Typeface.create(Typeface.DEFAULT, Typeface.BOLD), 32, true, Color.BLACK);

            this.mEngine.getTextureManager().loadTexture(this.BTAblog);
            this.mEngine.getTextureManager().loadTexture(this.BTAtic);
            this.mEngine.getTextureManager().loadTexture(this.BTAChat);
            this.mEngine.getTextureManager().loadTexture(this.BTAdado);

      this.mEngine.getTextureManager().loadTexture(this.BTAFondoJuego);
            this.mEngine.getTextureManager().loadTexture(this.BTAcarta);
            this.mEngine.getTextureManager().loadTexture(this.BTAdorso);

            for (int p = 0; p < numero; p++) {

      this.mEngine.getTextureManager().loadTexture(this.BTAflags[p]);
            }


      this.mEngine.getTextureManager().loadTexture(this.mFontTextureGrande);

      this.mEngine.getTextureManager().loadTexture(this.mFontTextureChat);
            this.mEngine.getTextureManager().loadTexture(this.mFontTexture);

      this.mEngine.getTextureManager().loadTexture(this.mFontTextureBlue);

      this.mEngine.getTextureManager().loadTexture(this.mFontTexture2);
            this.mEngine.getFontManager().loadFont(this.mFontBlue);
            this.mEngine.getFontManager().loadFont(this.mFont);
            this.mEngine.getFontManager().loadFont(this.mFont2);
            this.mEngine.getFontManager().loadFont(this.mFontChat);
            this.mEngine.getFontManager().loadFont(this.mFontGrande);
      }

      publicboolean onKeyDown(int keyCode, KeyEvent event) {

            switch (keyCode) {

            case KeyEvent.KEYCODE_BACK:



                  this.runOnUiThread(new Runnable() {
                        @Override
                        publicvoid run() {
                              crearDialogoConfirmacion();
                        }
                  });
                  break;
```

```
            case KeyEvent.KEYCODE_MENU:


            default:
                  returnsuper.onKeyDown(keyCode, event);

            }
            returntrue;
      }

      privatevoid crearDialogoConfirmacion() {

            AlertDialog.Builder dialog = new AlertDialog.Builder(this);

            dialog.setMessage("¿Salir?");
            dialog.setCancelable(false);
            dialog.setPositiveButton("Si", new
DialogInterface.OnClickListener() {

                  @Override
                  publicvoid onClick(DialogInterface dialog, int which) {
                        finish();
                  }
            });
            dialog.setNegativeButton("No", new
DialogInterface.OnClickListener() {

                  @Override
                  publicvoid onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                  }
            });
            dialog.show();

      }

      publicvoid tostar(final String texto) {
            final Game juego = this;

            this.runOnUiThread(new Runnable() {
                  publicvoid run() {
                        Toast.makeText(juego, texto,
Toast.LENGTH_LONG).show();
                  }
            });

      }

      publicvoid cargarChat() {

            if (SPRchat == null) {
                  SPRchat = new Sprite(0, 0, this.TRChat);
                  SPRchat.attachChild(new Sprite(0, 0,
this.TRChat_superior));
                  SPRchat.attachChild(new Sprite(0, 0, this.TRChat_back));
```

```
                  chat = new ChangeableText(20, 0, mFontChat, " ",
                              "nombre del pais mas largo nombre del pais
mas largo nombre del pais mas largo"
                                          .length()) {
                        publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent,
                                    finalfloat pTouchAreaLocalX,
                                    finalfloat pTouchAreaLocalY) {
                              System.out.println("tocado texto");
                              super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX,
                                          pTouchAreaLocalY);
                              returnfalse;
                        }
                  };
                  SPRchat.attachChild(chat);

                  SPRchat.setZIndex(1000);
                  this.getEngine().getScene().attachChild(SPRchat);
                  // this.escenaMenu.attachChild(SPRchat);
            }

            InputMethodManager m = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            if (m != null) {
                  m.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
                  // m.toggleSoftInput(0, InputMethodManager.SHOW_IMPLICIT);

            }

      }

      publicvoid cargarTimer() {
            final Scene scene = new Scene();
            finalintcenterX = (CAMERA_WIDTH - this.TTRtimer.getWidth()) / 2;
            finalintcenterY = (CAMERA_HEIGHT - this.TTRtimer.getHeight()) /
2;
            timer = new AnimatedSprite((800 - 160) / 2, (480 - 160) / 2,
TTRtimer);

            scene.attachChild(timer);
            timer.animate(80);
            this.mEngine.setScene(scene);
      }

      @Override
      public Scene onLoadScene() {


            final Scene scene = new Scene();
            finalintcenterX = (CAMERA_WIDTH - this.TTRtimer.getWidth()) / 2;
            finalintcenterY = (CAMERA_HEIGHT - this.TTRtimer.getHeight()) /
2;
            timer = new AnimatedSprite((800 - 160) / 2, (480 - 160) / 2,
TTRtimer);

            scene.attachChild(timer);
            timer.animate(80);
```

```
		return scene;
	}



	publicvoid finalizar (){

		this.runOnUiThread(new Runnable() {
			@Override
			publicvoid run() {
				finish();
			}
		});

	}
	@Override
	publicvoid onLoadComplete() {

		this.timer.registerEntityModifier(new Delay(1.1f) {
			protectedvoid onModifierStarted(IEntity pItem) {
				runOnUiThread(new Runnable() {

					@Override
					publicvoid run() {
						// TODO Auto-generated method stub

						carga();
					}

				});
			}

			protectedvoid onModifierFinished(IEntity pItem) {

			}
		});

	}

	publicvoid carga() {

		datos = new datosUsuario(this);
		datos.leer_config();
		System.out.println("leodatosusuario" + datos.getUsuario());
		Bundle bundle = getIntent().getExtras();
		System.out.println("b online"
				+
Boolean.parseBoolean(bundle.getString("online")));
		System.out.println("jugadores"
				+ Integer.parseInt(bundle.getString("jugadores")));
		switch (Integer.parseInt(bundle.getString("jugadores"))) {
		case 2:
			cagarPartidaLocal(2,

	Boolean.parseBoolean(bundle.getString("online")));
			break;
		case 3:
			cagarPartidaLocal(3,
```

```
	Boolean.parseBoolean(bundle.getString("online")));
			break;
		case 4:
			cagarPartidaLocal(4,

	Boolean.parseBoolean(bundle.getString("online")));
			break;
		case 21:
			cagarPartidaLocal(21,

	Boolean.parseBoolean(bundle.getString("online")));
			break;

		}

	}

	publicvoid cagarPartidaLocal(int numJugadores, boolean online) {
		// cargarTimer();

		this.online = online;
		Semaphore semaforo =new Semaphore(0);

		System.out.println("cagarBaraja()");
		this.CameraGame = new Camera3d(0, 0, CAMERA_WIDTH,
CAMERA_HEIGHT);
		CameraGame.set3d();
		// this.CameraGame.setZClippingPlanes(-100, 100);
		Log.i("this.CameraGame.getFarZClippingPlane()",

	Float.toString(this.CameraGame.getFarZClippingPlane()));

		escenaJuego = new CamaraEscena(CameraGame);
		System.out.println("cagarBaraja()1");
		// contadordefotogramas

		/*
		final FPSCounter fpsCounter = new FPSCounter();
		this.mEngine.registerUpdateHandler(fpsCounter);


		final ChangeableText textCenter = new ChangeableText(50, 400,
				this.mFont2, "Seconds elapsed:",
				"Seconds elapsed: XXXXX".length());

		escenaJuego.attachChild(textCenter);

		escenaJuego.registerUpdateHandler(new TimerHandler(1 / 20.0f,
true,
				new ITimerCallback() {
					@Override
					public void onTimePassed(final TimerHandler
pTimerHandler) {

						textCenter.setText("FPS: " + (int)
fpsCounter.getFPS());
					}
				}));
```

```
	*/


			System.out.println("cagarBaraja()2");
			Sprite fondo = new Sprite(0, 0, TRFondoJuego) {

				protectedvoid applyRotation(final GL10 pGL) {

	GLHelper.disableCulling(pGL);

					finalfloat rotation = this.mRotation;

					if (rotation != 0) {
						finalfloat rotationCenterX =
this.mRotationCenterX;
						finalfloat rotationCenterY =
this.mRotationCenterY;

						pGL.glTranslatef(rotationCenterX,
rotationCenterY, 0);

						pGL.glRotatef(rotation, 1, 0, 0);
						pGL.glTranslatef(-rotationCenterX, -
rotationCenterY, 0);
					}
				}

			};
			// fondo.setRotation(-18.5f);
			// fondo.setScale(1.1f);
			System.out.println("cagarBaraja()3");
			EntityBackground back = new EntityBackground(fondo);
			escenaJuego.setBackground(back);

			int z = 1;
			MiSprite spr;

			for (Iterator<Carta> it = baraja.getList().iterator();
it.hasNext(); z++) {
				Carta carta = (Carta) it.next();

				/*
				 * runOnUiThread(new Runnable() {
				 *
				 * @Override public void run() { // Call
AsyncTask.execute() in
				 * here. //timer.nextTile(); } });
				 */

				System.out.println("cagarBaraja()Iterator<Carta>");
				spr = new MiSprite(this, carta, 400, 0, TRcarta,
TRpaises[z - 1],
						TRdorso, this.mFont, mFontBlue, z);
				spr.setZIndex(z);
				spr.setVisible(false);
				// spr.registrar();
				getSprites().add(spr);
```

```
                carta.setSprite(spr);
                spr.setScale(0.2f);
                spr.verDorso();

        }
        escenaJuego.sortChildren();

        tactil = new DirectorTactil(this);
        escenaJuego.setOnSceneTouchListener(tactil);
        escenaJuego.sortChildren();

        dado = new Dado(300, 150, this.TTRdado, Sonido_Dados);
        dado.setVisible(false);
        dado.setScale(3);
        escenaJuego.attachChild(dado);

        blog = new BlogSprite(this, 50, 50, this.TRblog);
        blog.setVisible(false);
        escenaJuego.attachChild(blog);
        // escenaMenu.registerTouchArea(blog);

        tic = new TicSprite(this, 600, 200, TTRtic);

        escenaJuego.attachChild(tic);

        directorjuego = new DirectorJuego(this, dado, escenaJuego);
        this.mEngine.setScene(escenaJuego);

        if (numJugadores == 21)directorjuego.verCartas(baraja);// este
        elsedirectorjuego.nuevaPartida(numJugadores, online, baraja);
}

@Override
publicvoid onResumeGame() {
        // TODO Auto-generated method stub
        System.out.println("resumeeee");
        super.onResumeGame();
}

@Override
publicvoid onPauseGame() {
        // TODO Auto-generated method stub
        System.out.println("pausee");
        super.onPauseGame();
}

protectedvoid onPause() {
        super.onPause();
        // this.overridePendingTransition(0, 0);
}

@Override
publicvoid onWindowFocusChanged(finalboolean pHasWindowFocus) {
finalboolean dialogShowing = true;

super.onWindowFocusChanged(pHasWindowFocus || dialogShowing);
}
```

Archivo MainMenu.java

```
package xnetcom.pro.cartas.activities;

import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.engine.options.EngineOptions;
import org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import
org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy;
import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.ColorBackground;
import org.anddev.andengine.entity.scene.menu.MenuScene;
import
org.anddev.andengine.entity.scene.menu.MenuScene.IOnMenuItemClickListener;
import org.anddev.andengine.entity.scene.menu.item.IMenuItem;
import org.anddev.andengine.entity.sprite.Sprite;
importorg.anddev.andengine.entity.util.FPSLogger;
import org.anddev.andengine.opengl.font.Font;
import org.anddev.andengine.opengl.font.FontFactory;
importorg.anddev.andengine.opengl.texture.Texture;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlas;
import
org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlasTextureReg
ionFactory;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.ui.activity.BaseGameActivity;

importxnetcom.pro.cartas.R;
import xnetcom.pro.cartas.auxiliares.Menus;
import xnetcom.pro.cartas.auxiliares.datosUsuario;
import xnetcom.pro.cartas.sprites.Delay;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.Typeface;
import android.widget.FrameLayout;
import android.widget.Toast;

publicclass MainMenu extends BaseGameActivity implements
IOnMenuItemClickListener {


	// ===========================================================
	// Constants
	// ===========================================================

	publicstaticfinalintCAMERA_WIDTH = 800;
	publicstaticfinalintCAMERA_HEIGHT = 480;


	privatestaticfinalintFONT_SIZE = 48;


	protectedstaticfinalintMENU_GAME1 = 0;
	protectedstaticfinalintMENU_OPTIONS = MENU_GAME1 + 3;
	protectedstaticfinalintMENU_QUIT = MENU_GAME1 + 4;
```

```
// ============================================================
// Fields
// ============================================================

protected Camera mCamera;

protected Scene mMainScene;

private BitmapTextureAtlas mFontTexture;
private BitmapTextureAtlas BTAportada;
private TextureRegion TRportada;
private BitmapTextureAtlas BTAFondoMenu;
private TextureRegion TRFondoMenu;
private Font mFont;
private Sprite splash;
private Menus menus;
private Font mFontGrande;
protected MenuScene mMenuScene;
private BitmapTextureAtlas mFontTextureGrande;
private datosUsuario datos;



/// para el tutorial

private BitmapTextureAtlas BTAsiguente;
private BitmapTextureAtlas BTAatras;
private BitmapTextureAtlas BTAsalir;
private BitmapTextureAtlas BTAfondo;

private TextureRegion TRsiguente;
private TextureRegion TRatras;
private TextureRegion TRsalir;
private TextureRegion TRfondo;



public Menus getMenus(){
	returnmenus;
}

public TextureRegion getTRfondo(){
	returnthis.TRfondo;

}

public TextureRegion getTRsiguente() {
	returnTRsiguente;
}

publicvoid setTRsiguente(TextureRegion tRsiguente) {
	TRsiguente = tRsiguente;
}

public TextureRegion getTRatras() {
	returnTRatras;
```

```
	}

	publicvoid setTRatras(TextureRegion tRatras) {
		TRatras = tRatras;
	}

	public TextureRegion getTRsalir() {
		returnTRsalir;
	}

	publicvoid setTRsalir(TextureRegion tRsalir) {
		TRsalir = tRsalir;
	}

	publicvoid cargarTutorial(){
		BTAsiguente = new BitmapTextureAtlas(128, 128,
TextureOptions.NEAREST);
		BTAatras = new BitmapTextureAtlas(128, 128,
TextureOptions.NEAREST);
		BTAsalir = new BitmapTextureAtlas(128, 128,
TextureOptions.NEAREST);
		BTAfondo = new BitmapTextureAtlas(1024, 512,
TextureOptions.NEAREST);

		BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");

		this.TRsiguente =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAsiguente,
this,"flecha_derecha.png", 0, 0);
		this.TRatras =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAatras,
this,"flecha_iaquierda.png", 0, 0);
		this.TRsalir =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAsalir,
this,"flecha_salida.png", 0, 0);
		this.TRfondo=
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAfondo, this,
"tuto_0.png", 0, 0);

		this.mEngine.getTextureManager().loadTexture(this.BTAsiguente);
		this.mEngine.getTextureManager().loadTexture(this.BTAatras);
		this.mEngine.getTextureManager().loadTexture(this.BTAsalir);
		this.mEngine.getTextureManager().loadTexture(this.BTAfondo);
	}
	// sirveparacambier el tutorial a otroestado
	publicvoid cambiaTuto(int num){
		this.TRfondo=
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAfondo, this,
"tuto_"+num+".png", 0, 0);
		///this.mEngine.getTextureManager().loadTexture(this.BTAfondo);


	}

	public datosUsuario getDatos(){
		returndatos;
	}
	publicvoid setDatos(datosUsuario datos){
		this.datos=datos;
```

```
	}
	@Override
	public Engine onLoadEngine() {
		System.out.println("onLoadEngine()");
		this.mCamera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);

		returnnew Engine(new EngineOptions(true,
ScreenOrientation.LANDSCAPE,
				new RatioResolutionPolicy(CAMERA_WIDTH,
CAMERA_HEIGHT), this.mCamera));
	}

	public FrameLayout.LayoutParams getLayoutParams(){
		returnsuper.createSurfaceViewLayoutParams();
	}
	@Override
	publicvoid onLoadResources() {
		System.out.println("onLoadResources()");

		this.BTAportada = new BitmapTextureAtlas(1024, 512,
TextureOptions.BILINEAR_PREMULTIPLYALPHA);
		this.mFontTexture= new BitmapTextureAtlas(256, 256,
TextureOptions.BILINEAR_PREMULTIPLYALPHA);
		BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");
		this.TRportada =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAportada, this,
"portada.png", 0, 0);
		this.mEngine.getTextureManager().loadTexture(this.BTAportada);

		 FontFactory.setAssetBasePath("font/");
this.mFont = FontFactory.createFromAsset(this.mFontTexture, this,
"Zrnic.ttf", FONT_SIZE, true, Color.BLACK);
this.mEngine.getTextureManager().loadTexture(this.mFontTexture);
//this.getFontManager().loadFont(this.mFont);
this.mEngine.getFontManager().loadFont(this.mFont);

	}


	privatevoid CargaRecursos(){
		datos=new datosUsuario(this);
		System.out.println("CargaRecursos()");
		BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/");
		this.BTAFondoMenu = new BitmapTextureAtlas(1024, 512,
TextureOptions.NEAREST);
		this.mFontTextureGrande = new BitmapTextureAtlas(512, 256,
TextureOptions.BILINEAR_PREMULTIPLYALPHA);

		this.TRFondoMenu =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.BTAFondoMenu,
this, "fondo_menu.png", 0, 0);
		this.mFontGrande = new Font(this.mFontTextureGrande,
Typeface.create(Typeface.DEFAULT, Typeface.NORMAL), 48, true, Color.BLACK);
		this.mEngine.getTextureManager().loadTexture(this.BTAFondoMenu);

	this.mEngine.getTextureManager().loadTexture(this.mFontTextureGrande);
		//this.getFontManager().loadFont(this.mFontGrande);
		this.mEngine.getFontManager().loadFont(this.mFontGrande);
```

```
}



@Override
public Scene onLoadScene() {
	System.out.println("onLoadScene())");
	final Scene scene = new Scene();
	splash = new Sprite(0, 0, this.TRportada);
	scene.setBackground(new ColorBackground(0, 0, 0));
	scene.attachChild(splash);
	return scene;
}

@Override
publicvoid onLoadComplete() {
	//carga();
	System.out.println("onLoadComplete()");

	this.splash.registerEntityModifier(new Delay(2f){
		protectedvoid onModifierFinished(IEntity pItem) {
			carga();

		}
	});

}


publicvoid carga(){
	System.out.println("carga()");
	CargaRecursos();
	//datos=new datosUsuario(this);
	menus = new Menus(this);
	menus.cargaMenuPrincipal();
}

publicvoid cagarPartida(int numjug, boolean online) {
	System.out.print("soy mainmenuonline"+online);

	String linea = new Boolean(online).toString();
	Intent juego=new Intent(MainMenu.this, Game.class);
	switch (numjug) {
	case 2:
		juego.putExtra("jugadores", "2");//pasarparametros
		juego.putExtra("online", linea);//pasarparametros
		startActivity(juego);
		break;
	case 3:
		juego.putExtra("jugadores", "3");	//pasarparametros
		juego.putExtra("online", linea);//pasarparametros
		startActivity(juego);
		break;
	case 4:
		juego.putExtra("jugadores", "4");	//pasarparametros
		juego.putExtra("online", linea);//pasarparametros
		startActivity(juego);
		break;
	case 21:
```

```
                juego.putExtra("jugadores", "21");     //pasarparametros
                juego.putExtra("online", linea);//pasarparametros
                startActivity(juego);
                break;
            }

        }



        @Override
        publicboolean onMenuItemClicked(final MenuScene pMenuScene, final
IMenuItem pMenuItem, finalfloat pMenuItemLocalX, finalfloat pMenuItemLocalY)
{
            /*
            switch(pMenuItem.getID()) {
                case MENU_GAME1:

                    Intent intent1=new Intent(MainMenu.this,
Game.class);
                    startActivity(intent1);
                    return true;
                case MENU_OPTIONS:

                    //Intent intent4=new Intent(MainMenu.this,
Options.class);
                    //startActivity(intent4);
                    return true;
                case MENU_QUIT:

                    this.finish();
                    return true;
                default:
                    return false;
            }
            */
            returnfalse;
        }


        publicvoid tostar(final String texto) {
            final BaseGameActivity menu=this;

            this.runOnUiThread(new Runnable() {
            publicvoid run() {
                Toast.makeText(menu, texto, Toast.LENGTH_LONG).show();
              }
            });

        }



        public TextureRegion getFondoMenu(){
            returnTRFondoMenu;
        }

        public Font getFontGrande() {
```

```
            returnmFontGrande;
      }


      public Font getFont() {
            returnmFont;
      }



      //Transitions done when an activity is pushed to the background
      @Override
      protectedvoid onPause() {
            System.out.println("onPause()");
            super.onPause();
            //MainMenu.this.overridePendingTransition(R.anim.push_left_in,
R.anim.push_left_out);
      }
}
```

Archivo CajaTexto.java

```
package xnetcom.pro.cartas.auxiliares;

import org.anddev.andengine.ui.activity.BaseGameActivity;

import xnetcom.pro.cartas.activities.MainMenu;
import android.app.AlertDialog;
import android.text.InputType;
import android.view.Gravity;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.content.DialogInterface;
import android.content.DialogInterface.OnShowListener;
import android.text.method.PasswordTransformationMethod;
import android.os.SystemClock;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout.LayoutParams;
import android.widget.TableLayout;

publicclass cajatexto{



	private String mValue="valorini";

	BaseGameActivity contexto;
	public cajatexto (BaseGameActivity contexto){
		this.contexto=contexto;
	}

	publicvoid accion(){
		//menu.cagarBaraja();
	}



	publicvoid asd(){
		contexto.runOnUiThread(new Runnable() {

@Override
publicvoid run() {
			AlertDialog.Builder alert = new
AlertDialog.Builder(contexto);

			alert.setTitle("HIGHSCORE");
//  alert.setIcon(R.drawable.trophy);
			alert.setMessage("Score : "+"\nEnter your name:");

final LinearLayout layout = new LinearLayout(contexto);
final EditText input = new EditText(contexto);
			layout.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));
			input.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));
			layout.setPadding(20, 0, 20, 0);
```

```
                    layout.addView(input);

                    alert.setView(layout);

                    alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
publicvoid onClick(DialogInterface dialog, int whichButton) {
                      String value = input.getText().toString();

//highscore.addScore(value,score.getScore());
// finish();
                      }
                    });

                    alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
publicvoid onClick(DialogInterface dialog, int whichButton) {
//  finish();
                      }
                    });

                    alert.show();
            }
    });
      }

      publicvoid jo(){

            contexto.addContentView(editText(), new
ViewGroup.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));



            /*
            EditText chatEditText = new EditText(menu);
            chatEditText.setSingleLine(true);
            chatEditText.setId(1);
            chatEditText.setHint(" ");

            TableLayout tableLayout = new TableLayout(menu);
            tableLayout.setVerticalGravity(Gravity.BOTTOM);

            tableLayout.setLayoutParams(new
FrameLayout.LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));

            tableLayout.setHorizontalGravity(Gravity.CENTER_HORIZONTAL);
            tableLayout.setPadding(25, 25, 25, 308);
            tableLayout.addView(chatEditText);
            //LayoutParams lp = new
ViewGroup.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT);
            TableLayout.LayoutParams lp = new TableLayout.LayoutParams(
               ViewGroup.LayoutParams.FILL_PARENT,
               ViewGroup.LayoutParams.FILL_PARENT);
            menu.addContentView(tableLayout, lp);
            tableLayout.setVisibility(View.GONE);
            */
```

```
	}

	EditText editText;
	MotionEvent motionEvent ;
	publicvoid dd(){
		editText.setVisibility(View.INVISIBLE);
		long downTime = SystemClock.uptimeMillis();
		long eventTime = SystemClock.uptimeMillis() + 100;
		float x = 0.0f;
		float y = 0.0f;
		// List of meta states found here:
developer.android.com/reference/android/view/KeyEvent.html#getMetaState()
		int metaState = 0;
		motionEvent = MotionEvent.obtain(
		    downTime,
		    eventTime,
		    MotionEvent.ACTION_UP,
		    x,
		    y,
		    metaState
		);
		editText.dispatchTouchEvent(motionEvent);
	}

	private TableLayout editText() {

editText = new EditText(contexto){

	publicboolean onTouchEvent(MotionEvent event)
	  {
		System.out.println("evento");

	returnsuper.onTouchEvent(event);
	  }


	 };
	 EditText editText1= new EditText(contexto);
	 editText1.setSingleLine(true);
	 editText1.setId(2);
editText1.setHint(" ");
	 editText1.setImeOptions(0x00000006);//paraque no
pasealsiguientetextosolito
editText.setImeOptions(0x00000006);//paraque no pasealsiguientetextosolito
editText.setSingleLine(true);
editText.setId(1);
editText.setHint(" ");

tableLayout = new TableLayout(contexto);
tableLayout.setVerticalGravity(Gravity.TOP);
tableLayout.setLayoutParams(new
FrameLayout.LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));
tableLayout.setHorizontalGravity(Gravity.CENTER_HORIZONTAL);
tableLayout.setPadding(100, 0, 100, 130);
tableLayout.setPadding(100, 100, 50, 0);
```

```
tableLayout.addView(editText);
tableLayout.addView(editText1);

returntableLayout;
     }



       TableLayout tableLayout;


      publicvoid esconder(){
            tableLayout.setVisibility(4);//hacer invisible //0 visible

       }

      publicvoid showTextInput() {
            contexto.runOnUiThread(new Runnable() {
                  @Override
                  publicvoid run() {



                        final AlertDialog.Builder alert = new
AlertDialog.Builder(contexto);

                        alert.setTitle("Usuario y Contraseña");
                        //alert.setMessage("df");

                        final EditText editText = new EditText(contexto);
                        final EditText editText2 = new EditText(contexto);
                        editText.setId(1);
                        editText2.setId(2);

                        editText.setTextSize(20f);
                        //editText.setText(mValue);
                        editText.setGravity(Gravity.CENTER_HORIZONTAL);

                        editText2.setTextSize(20f);
                        //editText2.setText(mValue);
                        editText2.setGravity(Gravity.CENTER_HORIZONTAL);

                  //    if (false)
editText.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
                        editText.setInputType(InputType.TYPE_CLASS_TEXT);
                        editText2.setInputType(
InputType.TYPE_TEXT_VARIATION_PASSWORD);
                        editText2.setTransformationMethod(new
PasswordTransformationMethod());


                        editText.setHint(" Usuario ");
                        editText2.setHint(" Contraseña ");
                        editText.setText("usuario");
                        TableLayout Ty =new TableLayout(contexto);
                        Ty.addView(editText );
                        Ty.addView(editText2);
                        alert.setView(Ty);
```

```
                        final MainMenu menu =(MainMenu)contexto;
                        alert.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                                @Override
                                publicvoid onClick(DialogInterface dialog,
int whichButton) {
                                        menu.getDatos().restaurar();

      menu.getDatos().setUsuario(editText.getText().toString());

      menu.getDatos().setPassword(editText2.getText().toString());
                                        menu.getDatos().guardar_config();
                                        menu.getMenus().verPerfil();
                                }
                        });

                        alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                                @Override
                                publicvoid onClick(DialogInterface dialog,
int whichButton) {

                                }
                        });

                        final AlertDialog dialog = alert.create();

                        dialog.setOnShowListener(new OnShowListener() {
                                @Override
                                publicvoid onShow(DialogInterface dialog) {
                                        //editText.requestFocus();
                                        //final InputMethodManager imm =
(InputMethodManager) menu.getSystemService(menu.INPUT_METHOD_SERVICE);
                                        //imm.showSoftInput(editText,
InputMethodManager.SHOW_IMPLICIT);
                                }
                        });
                        dialog.show();
                        editText.setHint(" Usuario ");
                        editText2.setHint(" Contraseña ");
                }
            });
      }

}
```

Archivo CameraEscena.java

```
package xnetcom.pro.cartas.auxiliares;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.entity.scene.CameraScene;

publicclass CamaraEscena extends CameraScene{


	public CamaraEscena() {
		super();
		// TODO Auto-generated constructor stub
	}

	public CamaraEscena(Camera pCamera) {
		super(pCamera);
		// TODO Auto-generated constructor stub
	}

	@Override
	protectedvoid onManagedDraw(final GL10 pGL, final Camera pCamera) {
		if(this.mCamera != null) {
			pGL.glMatrixMode(GL10.GL_PROJECTION);
			this.mCamera.onApplyCameraSceneMatrix(pGL);
			{
				pGL.glMatrixMode(GL10.GL_MODELVIEW);
				//pGL.glPushMatrix();                  //
comentadospara resolver el problemadeladescolocaciondeobjetos
				pGL.glLoadIdentity();

				super.onManagedDraw(pGL, mCamera);

				pGL.glPopMatrix();
			}
			pGL.glMatrixMode(GL10.GL_PROJECTION);
		}
	}

}
```

Archivo Camara3D.java

```
package xnetcom.pro.cartas.auxiliares;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.opengl.util.GLHelper;

import android.opengl.GLU;

publicclass Camera3d extends Camera
{

	public Camera3d(float pX, float pY, float pWidth, float pHeight) {
		super(pX, pY, pWidth, pHeight);
		// TODO Auto-generated constructor stub
	}


	privatebooleantipo=true;

	privatevoid setFrustum(GL10 pGL)
	   {

	// set field of view to 60 degrees
	float fov_degrees = 60;
	float fov_radians = fov_degrees / 180 * (float)Math.PI;

	// set aspect ratio and distance of the screen
	float aspect = this.getWidth() / this.getHeight();
	float camZ = this.getHeight()/2 / (float)Math.tan(fov_radians/2);

	// set projection
		GLHelper.setProjectionIdentityMatrix(pGL);
		GLU.gluPerspective(pGL, fov_degrees, aspect, camZ/10, camZ*10);

	// set view
		GLU.gluLookAt(pGL,   0,    0, camZ,   0,     0,       0,  0,
1, 0);

		pGL.glScalef(1,-1,1); // reverse y-axis
		pGL.glTranslatef(-800/2,-480/2,0); // origin at top left

	}

	publicvoid set3d(){
		tipo=true;
	   }
	publicvoid set2d(){
		tipo=false;
	   }

		@Override
		publicvoid onApplySceneBackgroundMatrix(GL10 pGL) {
			// TODO Auto-generated method stub
			if (tipo) setFrustum(pGL);
			elsesuper.onApplySceneBackgroundMatrix(pGL);
		}
```

```
		@Override
		publicvoid onApplyCameraSceneMatrix(GL10 pGL) {
			// TODO Auto-generated method stub
			if (tipo) setFrustum(pGL);
			elsesuper.onApplyCameraSceneMatrix(pGL);
		}
		publicvoid onApplySceneMatrix(GL10 pGL) {
			if (tipo) setFrustum(pGL);
			elsesuper.onApplySceneMatrix(pGL);
		}


}
```

Archivo DatosJugador.java

```
package xnetcom.pro.cartas.auxiliares;

publicclass DatosJugador {
	private String nombre;
	private String LvL;
	privateintbaraja[];

	publicvoid setNombre(String nombre) {
		this.nombre = nombre;
	}

	publicvoid setLvL(String LvL) {
		this.LvL = LvL;
	}

	publicvoid setBaraja(int[] baraja) {
		this.baraja = baraja;
	}

	public String getNombre() {
		returnnombre;
	}

	publicint[] getBaraja() {
		returnthis.baraja;
	}
}
```

Archivo DatosUsuario.java

```
package xnetcom.pro.cartas.auxiliares;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import org.anddev.andengine.ui.activity.BaseGameActivity;

import android.content.Context;
import android.util.Log;

publicclass datosUsuario {

	private String usuario;
	private String password;
	private String nombre;
	private String apellidos;
	private String pais;
	private String nivel;
	private String jugadas;
	private String ganadas;
	private String puntos;
	private BaseGameActivity juego;

	public datosUsuario(BaseGameActivity juego) {
		this.juego=juego;
		usuario = "default";
		password = "---";
		nombre = "---";
		apellidos = "---";
		pais = "---";
		nivel = "0";
		jugadas = "0";
		ganadas = "0";
		puntos = "0";

	}

	public String getUsuario() {
		returnusuario;
	}

	publicvoid restaurar(){
		usuario = "";
		password = "";
		nombre = "";
		apellidos = "";
		pais = "";
		nivel = "0";
		jugadas = "0";
		ganadas = "0";
		puntos = "0";
	}

	publicvoid setUsuario(String usuario) {
		this.usuario = usuario;
	}
```

```
public String getPassword() {
	returnpassword;
}

publicvoid setPassword(String password) {
	this.password = password;
}

public String getNombre() {
	returnnombre;
}

publicvoid setNombre(String nombre) {
	this.nombre = nombre;
}

public String getApellidos() {
	returnapellidos;
}

publicvoid setApellidos(String apellidos) {
	this.apellidos = apellidos;
}

public String getPais() {
	returnpais;
}

publicvoid setPais(String pais) {
	this.pais = pais;
}

public String getNivel() {
	returnnivel;
}

publicvoid setNivel(String nivel) {
	this.nivel = nivel;
}

public String getJugadas() {
	returnjugadas;
}

publicvoid setJugadas(String jugadas) {
	this.jugadas = jugadas;
}

public String getGanadas() {
	returnganadas;
}

publicvoid setGanadas(String ganadas) {
	this.ganadas = ganadas;
}

public String getPuntos() {
	returnpuntos;
```

```
	}

	publicvoid setPuntos(String puntos) {
		this.puntos = puntos;
	}

	publicvoid guardar_config() {
		try {
			OutputStreamWriter fout = new
OutputStreamWriter(juego.openFileOutput("config.ini", Context.MODE_PRIVATE));

			fout.write("usuario:"+getUsuario()+"\n");
			fout.write("password:"+getPassword()+"\n");
			fout.write("nombre:"+getNombre()+"\n");
			fout.write("apellidos:"+getApellidos()+"\n");
			fout.write("pais:"+getPais()+"\n");
			fout.write("nivel:"+getNivel()+"\n");
			fout.write("jugadas:"+getJugadas()+"\n");
			fout.write("ganadas:"+getGanadas()+"\n");
			fout.write("puntos:"+getPuntos()+"\n");
			fout.close();
		} catch (Exception ex) {
			Log.e("Ficheros", "Error al escribir fichero a memoria
interna");
		}

	}
	publicvoid leer_config(){
		try
		{
		    BufferedReader fin = new BufferedReader( new
InputStreamReader(juego.openFileInput("config.ini")));
		    String linea;
		while((linea = fin.readLine())!=null){
			System.out.println("linea: "+linea);
			String [] campos = linea.split(":");
			if(campos[0].equals("usuario")) setUsuario(campos[1]);
			if(campos[0].equals("password")) setPassword(campos[1]);
			if(campos[0].equals("nombre")) setNombre(campos[1]);
			if(campos[0].equals("apellidos")) setApellidos(campos[1]);
			if(campos[0].equals("pais")) setPais(campos[1]);
			if(campos[0].equals("nivel")) setNivel(campos[1]);
			if(campos[0].equals("jugadas")) setJugadas(campos[1]);
			if(campos[0].equals("ganadas")) setGanadas(campos[1]);
			if(campos[0].equals("puntos")) setPuntos(campos[1]);
		}

		    fin.close();
		}
		catch (Exception ex)
		{
		    Log.e("Ficheros", "Error al leer fichero desde memoria
interna");
		}
	}



}
```

Archivo Menus.java

```
package xnetcom.pro.cartas.auxiliares;

importjavax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.Engine;
importorg.anddev.andengine.engine.camera.Camera;
importorg.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.scene.CameraScene;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.EntityBackground;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.text.ChangeableText;
importorg.anddev.andengine.entity.text.Text;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlas;
import
org.anddev.andengine.opengl.texture.atlas.bitmap.BitmapTextureAtlasTextureReg
ionFactory;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;

import xnetcom.pro.cartas.activities.MainMenu;
import xnetcom.pro.cartas.red.HiloTCP;
import xnetcom.pro.cartas.sprites.SpriteBoton;

import android.graphics.Color;
import android.text.InputType;
import android.text.method.PasswordTransformationMethod;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.TableLayout;
import android.widget.TextView;
importandroid.widget.Toast;
import android.widget.RelativeLayout.LayoutParams;


publicclass Menus {
	private EditText edittext;
	private MainMenu menu;
	private CameraScene escenaMenu;
	private Engine engine;

	public Scene Menu_principal;
	private Scene Menu_Pindividual;
	private Scene Menu_Registro;
	private Scene Menu_Ponline;
	private Scene Menu_Perfil;

	private ChangeableText texto;
```

```
private SpriteBoton SBjugadores2;
private SpriteBoton SBjugadores3;
private SpriteBoton SBjugadores4;
private SpriteBoton SBatras;
private SpriteBoton SBayuda;
private SpriteBoton SBperfil;
private SpriteBoton SBcam_perfil;
private SpriteBoton SBmod_perfil;
private SpriteBoton SBcrearPerfil;
private SpriteBoton SBaceptar;
private SpriteBoton SBcancelar;
private SpriteBoton SBsalir;
private SpriteBoton SBcrearPartida;
private SpriteBoton SBunirsePartida;
private SpriteBoton SBverCartas;
private SpriteBoton SBonline;
private SpriteBoton SBpartida_ind;
private SpriteBoton SBpartida_multi;
private SpriteBoton SBblue;
private SpriteBoton SBtuto;
private SpriteBoton SBIatras;
private SpriteBoton SBIguardar;


private BitmapTextureAtlas BTAjugadores2;
private BitmapTextureAtlas BTAjugadores3;
private BitmapTextureAtlas BTAjugadores4;
private BitmapTextureAtlas BTAatras;
private BitmapTextureAtlas BTAayuda;
private BitmapTextureAtlas BTAperfil;
private BitmapTextureAtlas BTAcam_perfil;
private BitmapTextureAtlas BTAmod_perfil;
private BitmapTextureAtlas BTAver_perfil;
private BitmapTextureAtlas BTAcrearPerfil;
private BitmapTextureAtlas BTAaceptar;
private BitmapTextureAtlas BTAcancelar;
private BitmapTextureAtlas BTAsalir;
private BitmapTextureAtlas BTAcrearPartida;
private BitmapTextureAtlas BTAunirsePartida;
private BitmapTextureAtlas BTAverCartas;
private BitmapTextureAtlas BTAonline;
private BitmapTextureAtlas BTApartida_ind;
private BitmapTextureAtlas BTApartida_multi;
private BitmapTextureAtlas BTAblue;
private BitmapTextureAtlas BTAtuto;
private BitmapTextureAtlas BTAIatras;
private BitmapTextureAtlas BTAIguardar;


private TiledTextureRegion TTRjugadores2;
private TiledTextureRegion TTRjugadores3;
private TiledTextureRegion TTRjugadores4;
private TiledTextureRegion TTRatras;
private TiledTextureRegion TTRayuda;
private TiledTextureRegion TTRperfil;
private TiledTextureRegion TTRcam_perfil;
private TiledTextureRegion TTRmod_perfil;
private TiledTextureRegion TTRcrearPerfil;
private TiledTextureRegion TTRverPerfil;
```

```
	private TiledTextureRegion TTRaceptar;
	private TiledTextureRegion TTRcancelar;
	private TiledTextureRegion TTRsalir;
	private TiledTextureRegion TTRcrearPartida;
	private TiledTextureRegion TTRunirsePartida;
	private TiledTextureRegion TTRverCartas;
	private TiledTextureRegion TTRonline;
	private TiledTextureRegion TTRpartida_ind;
	private TiledTextureRegion TTRpartida_multi;
	private TiledTextureRegion TTRblue;
	private TiledTextureRegion TTRtuto;
	private TiledTextureRegion TTRIatras;
	private TiledTextureRegion TTRIguardar;
	private TextureRegion TTRazul;

	private EditText ETusuario;
	private EditText ETnombre;
	private EditText ETapellidos;
	private EditText ETcontraseña;
	private EditText ETpais;
	private TableLayout tableLayout;

	private datosUsuario datos;
	privateintfuncionMenuRegistro=1;// parametroque dice
alregistrosicuandose pulse guardarespararegistro o modificacion

	private Tutorial tuturial;


	public EditText getETusuario(){
		returnETusuario;
	}

	public Menus(MainMenu menu){
		this.menu = menu;
		engine=menu.getEngine();
		datos= menu.getDatos();
		//cargaMenuPMultijugador();

		BTAjugadores2 = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAjugadores3 = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAjugadores4 = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAatras = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAayuda = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAperfil = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAcam_perfil = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAmod_perfil = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
		BTAcrearPerfil = new BitmapTextureAtlas(256,
128,TextureOptions.NEAREST);
		BTAver_perfil = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
```

```
            BTAaceptar = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAcancelar = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAsalir = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAcrearPartida = new BitmapTextureAtlas(256,
128,TextureOptions.NEAREST);
            BTAunirsePartida = new BitmapTextureAtlas(256, 128,
      TextureOptions.NEAREST);
            BTAverCartas = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAonline = new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTApartida_ind= new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTApartida_multi= new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAblue= new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAtuto= new BitmapTextureAtlas(256, 128,
TextureOptions.NEAREST);
            BTAIatras= new BitmapTextureAtlas(128, 128,
TextureOptions.NEAREST);
            BTAIguardar= new BitmapTextureAtlas(128, 128,
TextureOptions.NEAREST);



      BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("gfx/botones/"
);

            TTRjugadores2=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAjugadores
2, menu, "2_jugadores_tiled.png", 0, 0, 1, 2);
            TTRjugadores3=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAjugadores
3, menu, "3_jugadores_tiled.png", 0, 0, 1, 2);
            TTRjugadores4=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAjugadores
4, menu, "4_jugadores_tiled.png", 0, 0, 1, 2);
            TTRatras=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAatras,
menu, "atras_tiled.png", 0, 0, 1, 2);
            TTRayuda=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAayuda,
menu, "ayuda_tiled.png", 0, 0, 1, 2);
            TTRperfil=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAperfil,
menu, "perfil_tiled.png", 0, 0, 1, 2);
            TTRcam_perfil=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAcam_perfi
l, menu, "cambiar_perfil_tiled.png", 0, 0, 1, 2);
            TTRmod_perfil=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAmod_perfi
l, menu, "modificar_perfil_tiled.png", 0, 0, 1, 2);
            TTRverPerfil=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAver_perfi
l, menu, "ver_perfil_tiled.png", 0, 0, 1, 2);
```

```
            TTRcrearPerfil=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAcrearPerf
il, menu, "crear_perfil_tiled.png", 0, 0, 1, 2);
            TTRaceptar=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAaceptar,
menu, "aceptar_tiled.png", 0, 0, 1, 2);
            TTRcancelar=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAcancelar,
menu, "cancelar_tiled.png", 0, 0, 1, 2);
            TTRsalir=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAsalir,
menu, "salir_tiled.png", 0, 0, 1, 2);
            TTRcrearPartida=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAcrearPart
ida, menu, "crear_partida_tiled.png", 0, 0, 1, 2);
            TTRunirsePartida=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAunirsePar
tida, menu, "unirse_a_una_partida_tiled.png", 0, 0, 1, 2);
            TTRverCartas=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAverCartas
, menu, "ver_cartas_tiled.png", 0, 0, 1, 2);
            TTRonline=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAonline,
menu, "online_tiled.png", 0, 0, 1, 2);
            TTRpartida_ind=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTApartida_i
nd, menu, "partida_individual_tiled.png", 0, 0, 1, 2);
            TTRpartida_multi=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTApartida_m
ulti, menu, "partida_multijugador_tiled.png", 0, 0, 1, 2);
            TTRblue=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAblue,
menu, "bluetooth_tiled.png", 0, 0, 1, 2);
            TTRtuto=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAblue,
menu, "tutorial_tiled.png", 0, 0, 1, 2);
            TTRIatras=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAIatras,
menu, "icono_atras.png", 0, 0, 1, 2);
            TTRIguardar=
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.BTAIguardar,
menu, "icono_guardar.png", 0, 0, 1, 2);



      menu.getEngine().getTextureManager().loadTexture(this.BTAjugadores2);

      menu.getEngine().getTextureManager().loadTexture(this.BTAjugadores3);

      menu.getEngine().getTextureManager().loadTexture(this.BTAjugadores4 );
            menu.getEngine().getTextureManager().loadTexture(this.BTAatras);
            menu.getEngine().getTextureManager().loadTexture(this.BTAayuda);

      menu.getEngine().getTextureManager().loadTexture(this.BTAperfil);

      menu.getEngine().getTextureManager().loadTexture(this.BTAcam_perfil);

      menu.getEngine().getTextureManager().loadTexture(this.BTAver_perfil);
```

```
	menu.getEngine().getTextureManager().loadTexture(this.BTAmod_perfil);

	menu.getEngine().getTextureManager().loadTexture(this.BTAcrearPerfil);

	menu.getEngine().getTextureManager().loadTexture(this.BTAaceptar);

	menu.getEngine().getTextureManager().loadTexture(this.BTAcancelar);
		menu.getEngine().getTextureManager().loadTexture(this.BTAsalir);

	menu.getEngine().getTextureManager().loadTexture(this.BTAcrearPartida)
;

	menu.getEngine().getTextureManager().loadTexture(this.BTAunirsePartida
);

	menu.getEngine().getTextureManager().loadTexture(this.BTAverCartas);

	menu.getEngine().getTextureManager().loadTexture(this.BTAonline);

	menu.getEngine().getTextureManager().loadTexture(this.BTApartida_ind);

	menu.getEngine().getTextureManager().loadTexture(this.BTApartida_multi
);
		menu.getEngine().getTextureManager().loadTexture(this.BTAblue);
		menu.getEngine().getTextureManager().loadTexture(this.BTAtuto);

	menu.getEngine().getTextureManager().loadTexture(this.BTAIatras);

	menu.getEngine().getTextureManager().loadTexture(this.BTAIguardar);




	}



	public Engine getEngine() {
		returnengine;
	}
	publicvoid setEngine(Engine engine) {
		this.engine = engine;
	}
	publicvoid Perfil(){
		if(Menu_Perfil==null){
			Menu_Perfil= new Scene();
			Menu_Perfil.setBackground( new EntityBackground (new
Sprite(0,0,menu.getFondoMenu())));



			SpriteBoton SBver= new SpriteBoton(menu, 85, 100,
TTRverPerfil){
				publicvoid accion(){
					verPerfil();
				}
			};
```

```
                    SpriteBoton SBcam_perfil= new SpriteBoton(menu, 85, 200,
TTRcam_perfil){
                          publicvoid accion(){
                                cambiarPerfil();
                          }
                    };

                    SpriteBoton SBatras= new SpriteBoton(menu, 85, 300,
TTRatras){
                          publicvoid accion(){

                                cargaMenuPrincipal();
                          }
                    };

                    SpriteBoton SBmod_perfil= new SpriteBoton(menu,  465, 100,
TTRmod_perfil){
                          publicvoid accion(){
                                modificarPerfil();
                          }
                    };

                    SpriteBoton SBcrear_perfil= new SpriteBoton(menu,  465,
200, TTRcrearPerfil){
                          publicvoid accion(){
                                crearPerfil();
                          }
                    };



                    Menu_Perfil.attachChild(SBcrear_perfil);
                    Menu_Perfil.registerTouchArea(SBcrear_perfil);

                    Menu_Perfil.attachChild(SBver);
                    Menu_Perfil.registerTouchArea(SBver);
                    Menu_Perfil.attachChild(SBcam_perfil);
                    Menu_Perfil.registerTouchArea(SBcam_perfil);
                    Menu_Perfil.attachChild(SBmod_perfil);
                    Menu_Perfil.registerTouchArea(SBmod_perfil);
                    Menu_Perfil.attachChild(SBatras);
                    Menu_Perfil.registerTouchArea(SBatras);
              }
              engine.setScene(Menu_Perfil);

        }
        HiloTCP cliente=null;


        publicvoid verPerfil(){
              funcionMenuRegistro=1;
              cargaMenuRegistro(false,1);
              datos.leer_config();
              if (cliente==null){
                    cliente= new HiloTCP(menu);
                    cliente.start();
```

```
	cliente.getclienteTCP().enviarCLI_MOSTRAR_PERFIL(datos.getUsuario(),da
tos.getPassword());

			}else{
				if(cliente.isAlive()){
					System.out.println("icliente.isAlive()");
					cliente.desconectar();
					try {
						cliente.join();
					} catch (InterruptedException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
					}
				}
				cliente= new HiloTCP(menu);
				cliente.start();

	cliente.getclienteTCP().enviarCLI_MOSTRAR_PERFIL(datos.getUsuario(),da
tos.getPassword());
			}
	}

	privateclasshiloenvioTCP{

	}

	publicvoid cambiarPerfil(){
		cajatexto caja=new cajatexto(menu);
		caja.showTextInput() ;
	}




	publicvoid modificarPerfil(){
		funcionMenuRegistro=2;
		cargaMenuRegistro(true,2);

		//getETusuario().setText("asdsadadssdad");
		//cliente.getclienteTCP().cerrarConexion();


	}
	publicvoid crearPerfil(){
		funcionMenuRegistro=3;
		datos.restaurar();
		cargaMenuRegistro(true,3);
	}

	// intfuncion, 1 mostrar, 2 modificar,3 crear
	publicvoid cargaMenuRegistro(boolean editable,finalint funcion){


			if(Menu_Registro==null){
				Menu_Registro= new Scene();
				Menu_Registro.setBackground( new EntityBackground (new
Sprite(0,0,menu.getFondoMenu())));
```

```
                SBIatras= new SpriteBoton(menu, 450, 400, TTRIatras){
                        publicvoid accion(){
                                //menu.cagarPartida(2,false);
                                tableLayout.setVisibility(View.INVISIBLE);
                                cargaMenuPrincipal();

                        }
                };

                SBIguardar= new SpriteBoton(menu, 600, 400, TTRIguardar){
                        publicvoid accion(){
                                //menu.cagarPartida(2,false);
                                guardardatos();
                        }
                };

                Menu_Registro.attachChild(SBIatras);
                Menu_Registro.registerTouchArea(SBIatras);
                Menu_Registro.attachChild(SBIguardar);
                Menu_Registro.registerTouchArea(SBIguardar);

        }

        cajasDeTexto(editable,true);
        SBIguardar.setVisible(editable);
        engine.setScene(Menu_Registro);
}

publicvoid guardardatos(){

        switch (funcionMenuRegistro){
        case 2://modificar

                if (cliente==null){
                        cliente= new HiloTCP(menu);
                        cliente.start();
                        cliente.getclienteTCP().enviarCLI_MODIFICAR(
                                        ETusuario.getText().toString(),

datos.getPassword(),//contraseñaantigua

ETcontraseña.getText().toString(),//nuevacontraseña
                                        ETnombre.getText().toString(),
                                        ETapellidos.getText().toString(),
                                        ETpais.getText().toString());

                } else {
                        if (cliente.isAlive()){
                                System.out.println("icliente.isAlive()");
                                cliente.desconectar();
                                try {
                                        cliente.join();
                                } catch (InterruptedException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                        cliente= new HiloTCP(menu);
```

```
                        cliente.start();
                        cliente.getclienteTCP().enviarCLI_MODIFICAR(
                                    ETusuario.getText().toString(),
datos.getPassword(),//contraseñaantigua
ETcontraseña.getText().toString(),//nuevacontraseña
                                    ETnombre.getText().toString(),
                                    ETapellidos.getText().toString(),
                                    ETpais.getText().toString());
                  }



                  break;
            case 3:

                  if (cliente==null){
                        cliente= new HiloTCP(menu);
                        cliente.start();
                        cliente.getclienteTCP().enviarCLI_REGISTRAR(
                                    ETusuario.getText().toString(),
                                    ETcontraseña.getText().toString(),
                                    ETnombre.getText().toString(),
                                    ETapellidos.getText().toString(),
                                    ETpais.getText().toString());

                  } else {
                        if (cliente.isAlive()){
                              System.out.println("icliente.isAlive()");
                              cliente.desconectar();
                              try {
                                    cliente.join();
                              } catch (InterruptedException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                              }
                        }
                        cliente= new HiloTCP(menu);
                        cliente.start();
                        cliente.getclienteTCP().enviarCLI_REGISTRAR(
                                    ETusuario.getText().toString(),
                                    ETcontraseña.getText().toString(),
                                    ETnombre.getText().toString(),
                                    ETapellidos.getText().toString(),
                                    ETpais.getText().toString());
                  }
                  /*
                  tableLayout.setVisibility(View.INVISIBLE);
                  cargaMenuPrincipal();
                  */
                  break;
            default:
                  break;

            }

}
publicvoid cargarTutorial(){
```

```
		if (this.tuturial==null){
			this.tuturial= new Tutorial(this.menu);
		}
		tuturial.verTutorial();

	}

	publicvoid cargaMenuPrincipal(){
		//cargaMenuPMultijugador();


		if(Menu_principal==null){


			SpriteBoton SBpartida_ind= new SpriteBoton(menu, 85, 100,
TTRpartida_ind){
				publicvoid accion(){
					cargaMenuPindividual();
				}
			};
			SpriteBoton SBpartida_multi= new SpriteBoton(menu, 85,
200, TTRpartida_multi){
				publicvoid accion(){
					//cargaMenuPonline();
				}
			};
			SpriteBoton SBtuto= new SpriteBoton(menu, 85, 300,
TTRtuto){
				publicvoid accion(){
					cargarTutorial();
				}

			};

			SpriteBoton SBperfil= new SpriteBoton(menu, 465, 100,
TTRperfil){
				publicvoid accion(){
					Perfil();                                //
				}

			};
			SpriteBoton SBverCartas= new SpriteBoton(menu, 465, 200,
TTRverCartas){
				publicvoid accion(){
					menu.cagarPartida(21, false);
					System.out.println("vercartaaaass");
				}

			};
			SpriteBoton SBsalir= new SpriteBoton(menu, 465, 300,
TTRsalir){
				publicvoid accion(){
					menu.finish();
				}
			};;


			Menu_principal = new Scene();
```

```
                    Menu_principal.setBackground( new EntityBackground (new
Sprite(0,0,menu.getFondoMenu())));

                    Menu_principal.attachChild(SBsalir);
                    Menu_principal.attachChild(SBperfil);
                    Menu_principal.attachChild(SBpartida_ind);
                    Menu_principal.attachChild(SBpartida_multi);
                    Menu_principal.attachChild(SBverCartas);
                    Menu_principal.attachChild(SBtuto);


                    Menu_principal.registerTouchArea(SBsalir);
                    Menu_principal.registerTouchArea(SBperfil);
                    Menu_principal.registerTouchArea(SBpartida_ind);
                    Menu_principal.registerTouchArea(SBpartida_multi);
                    Menu_principal.registerTouchArea(SBverCartas);
                    Menu_principal.registerTouchArea(SBtuto);

             }
             engine.setScene(Menu_principal);


      }

      publicvoid cargaMenuPindividual(){
             //cargaMenuPMultijugador();

             if (Menu_Pindividual==null){

                    SpriteBoton SBjugadores2= new SpriteBoton(menu, 270, 50,
TTRjugadores2){
                           publicvoid accion(){
                                  menu.cagarPartida(2,false);
                           }
                    };
                    SpriteBoton SBjugadores3= new SpriteBoton(menu, 270, 150,
TTRjugadores3){
                           publicvoid accion(){
                                  menu.cagarPartida(3, false);
                           }
                    };
                    SpriteBoton SBjugadores4= new SpriteBoton(menu, 270, 250,
TTRjugadores4){
                           publicvoid accion(){
                                  menu.cagarPartida(4, false);
                           }
                    };
                    SpriteBoton SBatras= new SpriteBoton(menu, 270, 350,
TTRatras){
                           publicvoid accion(){
                                  cargaMenuPrincipal();
                           }
                    };

                    Menu_Pindividual = new Scene();
                    Menu_Pindividual.setBackground( new EntityBackground (new
Sprite(0,0,menu.getFondoMenu())));

                    Menu_Pindividual.attachChild(SBjugadores2);
```

```
                  Menu_Pindividual.attachChild(SBjugadores3);
                  Menu_Pindividual.attachChild(SBjugadores4);
                  Menu_Pindividual.attachChild(SBatras);

                  Menu_Pindividual.registerTouchArea(SBjugadores2);
                  Menu_Pindividual.registerTouchArea(SBjugadores3);
                  Menu_Pindividual.registerTouchArea(SBjugadores4);
                  Menu_Pindividual.registerTouchArea(SBatras);
            }
            engine.setScene(Menu_Pindividual);

      }

      publicvoid cargaMenuPonline(){
            //menu.cagarPartida(2,true);
            //cajasDeTexto();
            if (Menu_Ponline==null){
                  SpriteBoton SBatras= new SpriteBoton(menu, 270, 350,
TTRatras){
                        publicvoid accion(){
                              //tableLayout.setVisibility(View.INVISIBLE);
                              cargaMenuPrincipal();
                        }
                  };
                  Menu_Ponline = new Scene();
                  Menu_Ponline.setBackground( new EntityBackground (new
Sprite(0,0,menu.getFondoMenu())));
                  //texto=new ChangeableText(270, 150, menu.getFontGrande(),
" ", 200);
                  //texto.setText("Buscandopartidas");
                  //texto.setPosition((800/2f)-texto.getWidth()/2f, 100);

                  Menu_Ponline.attachChild(SBatras);
                  Menu_Ponline.registerTouchArea(SBatras);

            }
            engine.setScene(Menu_Ponline);
            menu.cagarPartida(2,true);

            //menu.Conectar();

      }

publicvoid actualizaCajas(){
      menu.runOnUiThread(new Runnable() {

      @Override
      // to safely detach and re-attach the sprites
      publicvoid run() {
            cajasDeTexto(false, true);
      }
      });

}


      publicvoid cajasDeTexto(boolean editable, boolean visible){

            datos.leer_config();
```

```
            if (tableLayout==null){
                  tableLayout = new TableLayout(menu);
                  tableLayout.setVerticalGravity(Gravity.TOP);
                  tableLayout.setLayoutParams(new
FrameLayout.LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.FILL_PARENT));

      tableLayout.setHorizontalGravity(Gravity.CENTER_HORIZONTAL);
                  tableLayout.setPadding(0, 25, 0, 0);

                  ETusuario =new EditText(menu);
                  ETusuario.setSingleLine(true);
                  ETusuario.setId(1);
                  ETusuario.setHint("
");

                  ETusuario.setImeOptions(0x00000006);// paraque no
pasealsiguiente

      // textosolito

                  ETnombre = new EditText(menu);
                  ETnombre.setSingleLine(true);
                  ETnombre.setId(2);
                  ETnombre.setHint("
");
                  ETnombre.setImeOptions(0x00000006);// paraque no
pasealsiguiente
                                    // textosolito

                  ETapellidos = new EditText(menu);
                  ETapellidos.setSingleLine(true);
                  ETapellidos.setId(3);
                  ETapellidos.setHint("
");
                  ETapellidos.setImeOptions(0x00000006);// paraque no paseal
                                    // siguientetextosolito

                  ETcontraseña = new EditText(menu);
                  ETcontraseña.setSingleLine(true);
                  ETcontraseña.setId(4);
                  ETcontraseña.setHint("
");
                  ETcontraseña.setImeOptions(0x00000006);/

      ETcontraseña.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD);
                  ETcontraseña.setTransformationMethod(new
PasswordTransformationMethod());


                  ETpais = new EditText(menu);
                  ETpais.setSingleLine(true);
                  ETpais.setId(5);
                  ETpais.setHint("               ");
                  ETpais.setImeOptions(0x00000006);//paraque no
pasealsiguientetextosolito
```

```
                LinearLayout Ly =new LinearLayout(menu);

                TextView text = new TextView(menu);
                text.setText("Usuario: ");
                text.setTextColor(Color.parseColor("#000000"));
                text.setTextSize(30f);
                Ly.addView(text);
                Ly.addView(ETusuario);
                tableLayout.addView(Ly);

                LinearLayout Ly2 =new LinearLayout(menu);
                TextView text2 = new TextView(menu);
                text2.setText("Nombre: ");
                text2.setTextColor(Color.parseColor("#000000"));
                text2.setTextSize(30f);
                Ly2.addView(text2);
                Ly2.addView(ETnombre);
                tableLayout.addView(Ly2);

                LinearLayout Ly3 =new LinearLayout(menu);
                TextView text3 = new TextView(menu);
                text3.setText("Apellidos: ");
                text3.setTextColor(Color.parseColor("#000000"));
                text3.setTextSize(30f);
                Ly3.addView(text3);
                Ly3.addView(ETapellidos);
                tableLayout.addView(Ly3);


                LinearLayout Ly4 =new LinearLayout(menu);
                TextView text4 = new TextView(menu);
                text4.setText("Contraseña: ");
                text4.setTextColor(Color.parseColor("#000000"));
                text4.setTextSize(30f);
                Ly4.addView(text4);
                Ly4.addView(ETcontraseña);
                tableLayout.addView(Ly4);

                LinearLayout Ly5 =new LinearLayout(menu);
                TextView text5 = new TextView(menu);
                text5.setText("Pais: ");
                text5.setTextColor(Color.parseColor("#000000"));
                text5.setTextSize(30f);
                Ly5.addView(text5);
                Ly5.addView(ETpais);
                tableLayout.addView(Ly5);
                if (!visible)tableLayout.setVisibility(View.INVISIBLE);
                if (visible)tableLayout.setVisibility(View.VISIBLE);
                menu.addContentView(tableLayout, new
ViewGroup.LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));

            }else{
                if (!visible)tableLayout.setVisibility(View.INVISIBLE);
                if (visible)tableLayout.setVisibility(View.VISIBLE);
            }

            ETusuario.setText(datos.getUsuario());
```

```
            ETnombre.setText(datos.getNombre());
            ETapellidos.setText(datos.getApellidos());
            ETcontraseña.setText(datos.getPassword());
            ETpais.setText(datos.getPais());

            ETusuario.setFocusable(editable);
            ETusuario.setFocusableInTouchMode(editable);
            ETusuario.setClickable(editable);

            ETnombre.setFocusable(editable);
            ETnombre.setFocusableInTouchMode(editable);
            ETnombre.setClickable(editable);

            ETapellidos.setFocusable(editable);
            ETapellidos.setFocusableInTouchMode(editable);
            ETapellidos.setClickable(editable);

            ETcontraseña.setFocusable(editable);
            ETcontraseña.setFocusableInTouchMode(editable);
            ETcontraseña.setClickable(editable);

            ETpais.setFocusable(editable);
            ETpais.setFocusableInTouchMode(editable);
            ETpais.setClickable(editable);


        }


}
```

Archivo ParserBaraja.java

```
package xnetcom.pro.cartas.auxiliares;


import java.io.InputStream;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import xnetcom.pro.cartas.juego.Baraja;
import xnetcom.pro.cartas.juego.Carta;

publicclass Parser_Baraja {

	Document dom;
	DocumentBuilder builder;

	public Baraja parse(InputStream fraw) {


		DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
		try {
			builder = factory.newDocumentBuilder();
			dom = builder.parse(fraw);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}

		Element root = dom.getDocumentElement();
		NodeList paises = root.getElementsByTagName("pais");
		NodeList parametros;
		Baraja baraja=new Baraja();

		for (int i = 0; i < paises.getLength(); i++) {

			Carta carta= new Carta();
			Element pais=(Element) paises.item(i);
			parametros=pais.getChildNodes();
			carta.setNombre(pais.getAttribute("id"));

			for (int j = 0; j < parametros.getLength(); j++) {


	if(parametros.item(j).getNodeName().trim().equals("idh")){

	carta.setIdh(Float.parseFloat(parametros.item(j).getTextContent()));
				}

	if(parametros.item(j).getNodeName().trim().equals("pidh")){


	//System.out.println(parametros.item(j).getTextContent());
```

```
        carta.setPidh(Integer.parseInt(parametros.item(j).getTextContent()));
                            }

        if(parametros.item(j).getNodeName().trim().equals("salud")){

        //System.out.println(parametros.item(j).getNodeName().trim());


        carta.setSalud(Float.parseFloat(parametros.item(j).getTextContent()));
                            }

        if(parametros.item(j).getNodeName().trim().equals("educacion")){

        //System.out.println(parametros.item(j).getNodeName().trim());
                                    System.out.println("educacion
"+parametros.item(j).getTextContent());

        carta.setEducacion(Float.parseFloat(parametros.item(j).getTextContent(
)));
                            }

        if(parametros.item(j).getNodeName().trim().equals("ingresos")){

        //System.out.println(parametros.item(j).getNodeName().trim());

        carta.setIngresos(Float.parseFloat(parametros.item(j).getTextContent()
));
                            }

        if(parametros.item(j).getNodeName().trim().equals("continente")){
        carta.setContinente(parametros.item(j).getTextContent());

                            }

        if(parametros.item(j).getNodeName().trim().equals("archivo")){

        carta.setArchivo(parametros.item(j).getTextContent());

                            }

        if(parametros.item(j).getNodeName().trim().equals("numero")){

        //System.out.println(parametros.item(j).getNodeName().trim());

        carta.setNumero(Integer.parseInt(parametros.item(j).getTextContent()))
;
                            }
                    }
                    baraja.add(carta);
              }
        return baraja;
        }

}
```

Archivo Turorial.java

```
package xnetcom.pro.cartas.auxiliares;

import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.background.EntityBackground;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.input.touch.TouchEvent;


import xnetcom.pro.cartas.activities.MainMenu;


publicclass Tutorial{


	private Sprite siguente;
	private Sprite atras;
	private Sprite salir;
	private Sprite fondo;

	private Engine mEngine;
	private Scene ScenaAnterior;
	privateintnum_tuto=0;


	private Scene Scena_Tutorial;
	private MainMenu menu;

	public Tutorial(MainMenu menu){
		this.menu=menu;
		this.mEngine=menu.getEngine();
		ScenaAnterior=menu.getEngine().getScene();

		menu.cargarTutorial();

		fondo = new Sprite    (0, 0,menu.getTRfondo());

		salir = new Sprite(0, 380, menu.getTRsalir()) {
			publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent,
					finalfloat pTouchAreaLocalX, finalfloat
pTouchAreaLocalY) {
				switch (pSceneTouchEvent.getAction()) {

				case 1:
					salir();
					break;

				}
				returntrue;
			}
		};
		atras = new Sprite(90, 380, menu.getTRatras()) {
			publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent,
					finalfloat pTouchAreaLocalX, finalfloat
pTouchAreaLocalY) {
```

```
                        switch (pSceneTouchEvent.getAction()) {

                        case 1:
                              anterior();
                              break;

                        }
                        returntrue;
                  }
            };
            siguente = new Sprite(220, 380, menu.getTRsiguente()) {
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent,
                              finalfloat pTouchAreaLocalX, finalfloat
pTouchAreaLocalY) {
                        switch (pSceneTouchEvent.getAction()) {

                        case 1:
                              System.out.println("tocado siguente");
                                    siguente();

                              break;

                        }
                        returntrue;
                  }
            };

            Scena_Tutorial= new Scene();
            Scena_Tutorial.setBackground( new EntityBackground (fondo));
            Scena_Tutorial.attachChild(siguente);
            Scena_Tutorial.attachChild(atras);
            Scena_Tutorial.attachChild(salir);
            Scena_Tutorial.registerTouchArea(siguente);
            Scena_Tutorial.registerTouchArea(atras);
            Scena_Tutorial.registerTouchArea(salir);

      }
      publicvoid verTutorial(){
            mEngine.setScene(Scena_Tutorial);
      }

      publicvoid salir(){
            menu.getEngine().setScene(ScenaAnterior);
      }
      publicvoid siguente(){
            if (num_tuto!=7){
                  num_tuto++;
            }
            menu.cambiaTuto(num_tuto);
      }

      publicvoid anterior(){
            if (num_tuto!=0){
                  num_tuto--;
            }
            menu.cambiaTuto(num_tuto);
      }
}
```

Archivo Baraja.java

```
package xnetcom.pro.cartas.juego;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Random;
import java.util.Iterator;



publicclass Baraja {

	private Float mediaEDu= 0f;
	private Float mediaidh= 0f;
	private Float mediapib= 0f;
	private Float mediasalud= 0f;
	privateintpaises=0;
	privateintcartasporjugador=46;
	protected ArrayList<Carta>list;

	private Random generator;

	public Baraja(){
		list= new ArrayList<Carta>();
		generator = new Random();
	}

	public Baraja(ArrayList<Carta> list){
		this.list= list;
	}

	publicboolean add(Carta object) {
		// TODO Auto-generated method stub
		mediaEDu+=object.getEducacion();
		mediaidh+=object.getIdh();
		mediapib+=object.getIngresos();
		mediasalud+=object.getSalud();
		paises++;
		returnlist.add(object);
	}

	publicvoid medias(){
		System.out.println("media edu ="+(mediaEDu/paises));
		System.out.println("media idh ="+(mediaidh/paises));
		System.out.println("media pib ="+(mediapib/paises));
		System.out.println("media salud ="+(mediasalud/paises));
	}
	publicvoid clear() {
		// TODO Auto-generated method stub
		list.clear();
	}

	publicboolean contains(Carta object) {
		// TODO Auto-generated method stub
		returnlist.contains(object);
	}
```

```
	public Carta get(int index) {
		// TODO Auto-generated method stub
		returnlist.get(index);
	}
	public ArrayList<Carta> getList(){
		returnlist;
	}
	publicvoid reparte24(int jugadores){
		int cartas =list.size()/jugadores;

		HashSet<Carta> repartida=new HashSet<Carta>();


		boolean metido;
		for(int z=1;z<jugadores+1;z++){
			for(int i=0;i<cartas;i++){
				int numero=0;
				do{
					numero=tomaDecision(0, list.size()-1);
					//System.out.println("tomadecisionnumero
"+numero);

					metido=repartida.add(list.get(numero));
					//System.out.println("metido  "+metido);
				}while(!metido);
				list.get(numero).setJugador(z);
			}

		}


		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			//System.out.println("el jugadouir " +carta.getJugador()+"
tiene " + carta.getNombre());

		}

	}


	publicvoid reparteOnline(int jugadores, int[]...barajas ){
		ArrayList<Carta> listaTemp = new ArrayList<Carta>();

		System.out.println("barajas length j1"+barajas[0].length);
		System.out.println("barajas length j2"+barajas[1].length);
		for(int i=0;i<jugadores;i++){
			for(int j=0;j<barajas[0].length;j++){

	//System.out.println("barajas["+i+"]["+j+"]"+"="+(barajas[i][j])+"quee
slacarta"+list.get(barajas[i][j]-1).getNombre()+"y
numero"+list.get(barajas[i][j]-1).getNumero());


	System.out.println("barajas["+i+"]["+j+"]"+(barajas[i][j]));

				list.get(barajas[i][j]-
1).getSprite().setVisible(false);
```

```
                        list.get(barajas[i][j]-1).setJugador(i+1);
                        if(i==0)list.get(barajas[i][j]-
1).getSprite().registrar();
                        listaTemp.add(list.get(barajas[i][j]-1));
                  }
            }
            list = new ArrayList<Carta>();
            list=listaTemp;

            //comprobacion
            System.out.println("la lista tiene "+list.size());

            for ( Iterator<Carta> it = list.iterator();it.hasNext();){
                  Carta carta =(Carta)it.next();
                  System.out.println("222el jugador "
+carta.getNumJugador()+" tiene " + carta.getNombre());

            }
      }
            publicint[] reparteMerche(){

                        int aStart=1;
                        int aEnd=187;
                        int numcartas=187;
                        int[] retorno =newint[numcartas];
                        HashSet<Integer> repartida=new HashSet<Integer>();
                        boolean metido;
                        int aleatorio;
                        for(int i=0;i<numcartas;i++){

                              do{

                                          long range = (long) aEnd -
(long) aStart + 1;
                                    long fraction = (long) (range *
generator.nextDouble());
                                    aleatorio = (int) (fraction + aStart);

                                          metido=repartida.add(aleatorio);
                                      }while(!metido);
                                  retorno[i]=aleatorio;
                        }
                  return retorno;
              }
      publicvoid reparte(int jugadores){
            System.out.println("llamada a reparte");
            HashSet<String> repartida=new HashSet<String>();
            ArrayList<Carta> listaTemp = new ArrayList<Carta>();
            intmax=list.size();

            System.out.println("cartasporjugador " +cartasporjugador);
            boolean metido;

            for(int i=0; i<cartasporjugador;i++){
                  for(int z=1;z<jugadores+1;z++){
                        int index=-1;
                        do{
                              index=tomaDecision(0, list.size()-1);
```

```
	metido=repartida.add(list.get(index).getNombre());
                        }while(!metido);
                        list.get(index).getSprite().setVisible(false);
                        list.get(index).setJugador(z);
                        if(z==1)list.get(index).getSprite().registrar();
                        listaTemp.add(list.get(index));

                  }
            }
            list = new ArrayList<Carta>();
            list=listaTemp;



            //comprobacion
            System.out.println("la lista tiene "+list.size());

            for ( Iterator<Carta> it = list.iterator();it.hasNext();){
                  Carta carta =(Carta)it.next();
                  System.out.println("el jugador " +carta.getNumJugador()+"
tiene " + carta.getNombre());

            }
      }
      //repartelascartasparalaopciondevarcartas

      publicvoid reparteVerCartas(){
            System.out.println("llamada a reparte");
            HashSet<String> repartida=new HashSet<String>();
            ArrayList<Carta> listaTemp = new ArrayList<Carta>();
            intmax=list.size();

            System.out.println("cartasporjugador " +cartasporjugador);
            boolean metido;

            for(int i=0; i<187;i++){

                        int index=-1;
                        do{
                              index=186-i;

      metido=repartida.add(list.get(index).getNombre());
                        }while(!metido);
                        list.get(index).getSprite().setVisible(false);
                        list.get(index).setJugador(1);
                        list.get(index).getSprite().registrar();
                        listaTemp.add(list.get(index));


            }
            list = new ArrayList<Carta>();
            list=listaTemp;



            //comprobacion
            System.out.println("la lista tiene "+list.size());
```

```
		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			System.out.println("el jugador " +carta.getNumJugador()+"
tiene " + carta.getNombre());

		}
	}

	publicint getPosicion(Carta carta){
		returnlist.indexOf(carta);


	}
	publicint[] posiciones(Float salud, Float edu, Float pib){
		int possalud=1;
		int posedu=1;
		int pospib=1;
		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			if(salud<carta.getSalud()) possalud++;
			if(edu<carta.getEducacion()) posedu++;
			if(pib<carta.getIngresos()) pospib++;
		}
		System.out.println("possalud "+possalud);
		System.out.println("posedu "+posedu);
		System.out.println("pospib "+pospib);
		int salida[]={possalud,posedu,pospib};
		return salida;
	}

	// miraslosposibleserroresnuevafuncion 5/5/2012
	public Carta getCarta(int numCarta){
		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			if (carta.getNumero()==numCarta) return carta;

		}
		returnnull;
	}


	privatevoidseleccion(int seleccion){
		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			carta.setSeleccion(seleccion);
		}
	}
	//
	privateint tomaDecision(int aStart, int aEnd) {

long range = (long) aEnd - (long) aStart + 1;
long fraction = (long) (range * generator.nextDouble());
int aleatorio = (int) (fraction + aStart);
return aleatorio;
    }

}
```

Archivo Carta.java

```
package xnetcom.pro.cartas.juego;

import xnetcom.pro.cartas.jugadores.Jugador;
import xnetcom.pro.cartas.sprites.MiSprite;

publicclass Carta  implements Comparable<Carta>{


	private  String nombre;
	privateintpidh;
	privatefloatidh;
	privatefloatsalud;
	privatefloateducacion;
	privatefloatingresos;
	private  String continente;
	private  String archivo;
	privateintnumero;
	privateintjugador=0;
	private MiSprite sprite;
	privateintvalorSeleccion;
	privatebooleanordenarporZindeX;
	privateintsituacion=0;
	private Jugador player;

	/*tiposdesituacionenlasqueunacartapuedeestar
	 * 0 pordefectonadaenesperial no puedesertocada
	 * 1 unusuariola ha sacado y conuntoquesepuedeagrandar y alejar o
mover
	 * 2 setratadeunacartadeusuariohumanoqueestaendisposiciondequesellame
a lafunciondisponermazo
	 * 3 setratadeunacartaqueseencuentraensituaciondedisponermazo
	 * 4 setratadeunacartasepreparaparaqueseelijansusdatospararetar a
lasotrascartas
	 * 5 lacartaseencuentraenposicionparaserlacartaelegida
	 * 6 setratadelacartaseleccionadapor el jugador
	*
	 */

	public Carta(){}

	public Carta( String nombre, int pidh, float salud, float educacion,
float ingresos, String continente, String archivo, int numero ){

		this.nombre=nombre;
		this.pidh=pidh;
		this.salud=salud;
		this.ingresos=ingresos;
		this.educacion=educacion;
		this.continente=continente;
		this.archivo=archivo;
		this.numero=numero;

	}
	public MiSprite getSprite() {
		returnsprite;
	}
```

```
publicvoid setJugador(Jugador jugador){
	player=jugador;
}
public Jugador getJugador(){
	returnplayer;
}
public String getNombre() {
	returnnombre;
}
publicint getNumJugador() {
	returnjugador;
}
publicint getSituacion() {
	returnsituacion;
}
publicvoid setSituacion(int situacion) {
	this.situacion=situacion;
}
publicint getSeleccion() {
	returnvalorSeleccion;
}

publicint getPidh() {
	returnpidh;
}
publicfloat getSalud() {
	returnsalud;
}
publicfloat getIngresos() {
	returningresos;
}
publicfloat getIdh() {
	returnidh;
}
publicfloat getEducacion() {
	returneducacion;
}
public String getArchivo() {
	returnarchivo;
}

public String getContinente() {
	returncontinente;
}
publicint getNumero() {
	returnnumero;
}

publicvoid setSprite(MiSprite sprite) {
	this.sprite=sprite;
}

publicvoid setNombre(String nombre) {
	this.nombre=nombre;
}
publicvoid setPidh(int pidh) {
	this.pidh=pidh;
}
publicvoid setSeleccion(int valorSeleccion) {
```

```
        this.valorSeleccion=valorSeleccion;
        this.ordenarporZindeX=false;
}
publicvoid ordenarPorZindeX(){
        ordenarporZindeX=true;
}
publicfloat getValorSeleccion(int seleccion){
        switch (seleccion) {
        case 1:
                return getIdh();
        case 2:
                return getSalud();
        case 3:
                return  getEducacion();
        case 4:
                return getIngresos();
        }
        return 0;
}
publicvoid setIdh(float idh) {
        this.idh=idh;
}

publicvoid setSalud(float salud) {
        this.salud=salud;
}
publicvoid setIngresos(float ingresos) {
        this.ingresos=ingresos;
}
publicvoid setEducacion(float educacion) {
        this.educacion=educacion;
}
publicvoid setArchivo(String archivo) {
        this.archivo=archivo;
}
publicvoid setContinente(String continente) {
        this.continente=continente;
}
publicvoid setNumero(int numero) {
        this.numero=numero;
}
publicvoid setJugador(int numero) {
        jugador=numero;
}

@Override
publicint compareTo(Carta carta) {
        switch (valorSeleccion) {
        case 1:

                return comparaFloat(idh, carta.getIdh());
        case 2:

                return comparaFloat(salud, carta.getSalud());

        case 3:

                return comparaFloat(educacion, carta.getEducacion());
        case 4:
```

```
                //System.out.println("comparamo ingreosos");
                return comparaFloat(ingresos, carta.getIngresos());
            }
            return 0;
        }

        publicint comparaFloat(float primero, float segundo){

            if(primero==segundo) return 0;
            if(primero>segundo) return 1;
            if(primero<segundo) return -1;

            return 2;
        }

}
```

Archivo Director3D.java

```
package xnetcom.pro.cartas.juego;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;

import org.anddev.andengine.entity.IEntity;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.sprites.MiSprite;
import xnetcom.pro.cartas.sprites.MoverX;
import xnetcom.pro.cartas.sprites.MoverY;

publicclass Director3D {


	/*para z
	 * -1300		X		1700
	 *
	 *
	 * y
	 * 300
	 */
	intx = 0;
	inty = 1;
	Game game;
	intnumJugadores;

	int[] PosIni = {200, 150 };
	int[] PosJ1 = { 452, 200 };
	int[] PosJ2 = { -50, -100 };
	int[] PosJ3 = { 270, -100  };
	int[] PosJ4 = { 600, -100 };


	public Director3D(Game game) {
		this.game = game;
	}
	//recolocalascartasdelosjugadoresbotparaqueseanpequeñas y bajas el
zindex

	publicvoid recoloca(Carta carta){
		MiSprite sprt=carta.getSprite();

		if (sprt.getScaleX() >= 0.1f && sprt.getScaleX() < 0.3f){
			//izq
			if(sprt.getX()<100)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),10));

			//centro

	if(sprt.getX()>100&&sprt.getX()<400)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),274));

			//dere
```

```
            if(sprt.getX()>500)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),532));
            sprt.setZIndex(sprt.getZIndex()+1000);
            game.getScene().sortChildren();
            sprt.agrandar(0.5f);

            }

            if (sprt.getScaleX() >= 0.9f && sprt.getScaleX() < 1.1f){
                  //izq
                  if(sprt.getX()<100)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),-50));

                  //centro

      if(sprt.getX()>100&&sprt.getX()<500)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),274));

                  //dere
                  if(sprt.getX()>500)sprt.registerEntityModifier(new
MoverX(0.5f,sprt.getX(),600));
                  sprt.setZIndex(sprt.getZIndex()-1000);
                  game.getScene().sortChildren();
                  sprt.enpequenecer(0.5f);
            }


      }

      publicvoid reparteCartas(int numJugadores){


            Baraja baraja=game.getBaraja();

            this.numJugadores=numJugadores;
            //System.out.println("(baraja.getList().size()-numJugadores)
"+(baraja.getList().size()-numJugadores));
            int z=100;
            int i=1;
            for ( Iterator<Carta> it =
baraja.getList().iterator();it.hasNext();z++,i++){
                  Carta carta = (Carta)it.next();

                  MiSprite sprt =carta.getSprite();

                  int pos=carta.getNumJugador();

                  if(carta.getNumJugador()==1){
                        sprt.setZIndex(z+400);
                  }else sprt.setIndex(z);

                  if(i>(baraja.getList().size()-numJugadores)){
                        if(!sprt.hasParent())sprt.registrar();
                        sprt.setVisible(true);
                        sprt.setPosition(PosIni[x], PosIni[y]);
                  }else{
                        sprt.setPosition(posJugador(pos)[x],
posJugador(pos)[y]);
                  }
```

```
            }

            game.getScene().sortChildren();
            try {
                  Thread.currentThread().sleep(1000);
            } catch (InterruptedException e1) {
                  // TODO Auto-generated catch block
                  e1.printStackTrace();
            }
            //ahorarepartimoslascartasdelosjugadores
            z=100;
            i=1;
            for ( Iterator<Carta> it =
baraja.getList().iterator();it.hasNext();z++,i++){

                  Carta carta = (Carta)it.next();
                  MiSprite sprt =carta.getSprite();
                  int pos=carta.getNumJugador();

                  if(i>(baraja.getList().size()-numJugadores)){

                        //System.out.println("repartiendoultimos");
                        sprt.registerEntityModifier(new
MoverX(1f,sprt.getX(),posJugador(pos)[x]));
                        sprt.registerEntityModifier(new
MoverY(1f,sprt.getY(),posJugador(pos)[y]));


                        try {
                              Thread.currentThread().sleep(300);
                        } catch (InterruptedException e1) {
                              // TODO Auto-generated catch block
                              e1.printStackTrace();
                        }

                  }
            }

            Baraja bar=game.getDirectorJuego().gethumano().getBaraja();
            bar.get(bar.getList().size()-1).getSprite().setVisible(true);

            bar.get(bar.getList().size()-2).getSprite().setVisible(true);
            bar.get(bar.getList().size()-3).getSprite().setVisible(true);
            //bar.get(bar.getList().size()-4).getSprite().setVisible(true);

            game.getScene().sortChildren();
            try {
                  Thread.currentThread().sleep(2000);
            } catch (InterruptedException e1) {
                  // TODO Auto-generated catch block
                  e1.printStackTrace();
            }
      }



      publicvoid CartaAretar(Carta carta){
            carta.setSituacion(1);
```

```
            MiSprite sprt=carta.getSprite();
            sprt.setZIndex(sprt.getZIndex()+150);
            game.getScene().sortChildren();
            sprt.setVisible(true);
            sprt.registerEntityModifier(new MoverX(1f,sprt.getX(),250));
            sprt.registerEntityModifier(new MoverY(1f,sprt.getY(),150));
            //sprt.agrandar(2f);
            sprt.verCarta(2f);
      }

      publicvoid CartaAretarHumano(Carta carta){
            MiSprite sprt=carta.getSprite();
            sprt.enpequenecer(0.8f);
            sprt.registerEntityModifier(new MoverX(1f,sprt.getX(),350));
            sprt.registerEntityModifier(new MoverY(1f,sprt.getY(),150));
      }

      publicvoid EliminarCartas(Carta carta){

      }


      publicvoid sacarCarta(Carta carta){
            carta.setSituacion(1);
            MiSprite sprt=carta.getSprite();
            if(!sprt.hasParent())sprt.registrar();
            sprt.setZIndex(sprt.getZIndex()+150);
            game.getScene().sortChildren();
            sprt.setVisible(true);
            sprt.registerEntityModifier(new
MoverY(1f,sprt.getY(),sprt.getY()+100));
            sprt.verCarta(2f);
      }

      publicvoid sacarCartaBocaAbajo(Carta carta){
            carta.setSituacion(1);
            MiSprite sprt=carta.getSprite();
            if(!sprt.hasParent())sprt.registrar();
            sprt.setZIndex(sprt.getZIndex()+150);
            game.getScene().sortChildren();
            sprt.setVisible(true);
            sprt.registerEntityModifier(new
MoverY(1f,sprt.getY(),sprt.getY()+100));
            //sprt.verCarta(2f);
      }

      publicvoid disponerMazo(){
            Baraja baraja
=game.getDirectorJuego().getJugadores()[0].getBaraja();

            for(int i=0;
i<game.getDirectorJuego().getJugadores().length;i++){
                  try{

      game.getDirectorJuego().getJugadores()[i].getCartaSeleccionada().getSp
rite().recoloca();
                  }catch(Exception e){

                  }
```

```
            }
            ArrayList<Carta> list=baraja.getList();

            for ( Iterator<Carta> it = list.iterator();it.hasNext();){
                  Carta carta =(Carta)it.next();
                  carta.ordenarPorZindeX();
            }
            Collections.sort(list);

            if(list.size()<=6){
                  for ( Iterator<Carta> it = list.iterator();it.hasNext();){
                        Carta carta =(Carta)it.next();
                        MiSprite sprt=carta.getSprite();
                        sprt.setVisible(true);
                  }

            }else{
                  try {
                        list.get(list.size() -
1).getSprite().setVisible(true);
                        list.get(list.size() -
2).getSprite().setVisible(true);
                        list.get(list.size() -
3).getSprite().setVisible(true);
                  } catch (Exception e) {   }
                  try {
                        list.get(0).getSprite().setVisible(false);
                        list.get(1).getSprite().setVisible(false);
                        list.get(2).getSprite().setVisible(false);
                  } catch (Exception e) {   }
            }

            for (Iterator<Carta> it = list.iterator(); it.hasNext();) {
                  Carta carta = (Carta) it.next();
                  if (carta.getSituacion() == 2) {

                        MiSprite sprt = carta.getSprite();
                        sprt.agrandar(1f);
                        sprt.verCarta(1f);
                        sprt.registerEntityModifier(new MoverX(1f,
sprt.getX(),452) {
                              protectedvoid onModifierFinished(IEntity
pItem) {
                                    MiSprite df = (MiSprite) pItem;
                                    Carta carta = df.getCarta();
                                    carta.setSituacion(3);
                                    super.onModifierFinished(pItem);
                              }
                        });
                        sprt.registerEntityModifier(new MoverY(1f,
sprt.getY(),80));
                  }
            }

      }


      publicvoid ayudaMazo(){
            Baraja baraja =game.getDirectorJuego().gethumano().getBaraja();
```

```
		ArrayList<Carta> list=baraja.getList();
		if (list.size() >= 6) {
			for (Iterator<Carta> it = list.iterator(); it.hasNext();)
{
				Carta carta = (Carta) it.next();
				if (!it.hasNext())
					carta.getSprite().setVisible(false);
			}
		}
		try {
			list.get(list.size() - 1).getSprite().setVisible(true);
			list.get(list.size() - 2).getSprite().setVisible(true);
			list.get(list.size() - 3).getSprite().setVisible(true);
		} catch (Exception e) {   }


	}

	publicvoid recogerMazo3(){
		System.out.println("recogerMazo3()");
		Baraja baraja =game.getDirectorJuego().gethumano().getBaraja();
		ArrayList<Carta> list=baraja.getList();


		for ( Iterator<Carta> it = list.iterator();it.hasNext();){
			Carta carta =(Carta)it.next();
			if (carta.getSituacion()==3){
				carta.setSituacion(2);
				MiSprite sprt=carta.getSprite();
				sprt.enpequenecer(1f);
				sprt.verDorso(1f);

				sprt.registerEntityModifier(new
MoverX(0.2f,sprt.getX(),452));
				if(!it.hasNext()){
					sprt.registerEntityModifier(new
MoverY(0.2f,sprt.getY(),200){
						//el ultimoordena
						protectedvoid
onModifierFinished(IEntity pItem) {
							ayudaMazo();

	game.getScene().sortChildren();
							super.onModifierFinished(pItem);
						}

					});
				}else{
					sprt.registerEntityModifier(new
MoverY(0.2f,sprt.getY(),200));
				}


			}


		}
```

```
	}

	publicvoid recogerMazo(){

		Baraja baraja =game.getDirectorJuego().gethumano().getBaraja();
		ArrayList<Carta> list=baraja.getList();

		list.get(list.size()-1).getSprite().setVisible(true);
		boolean movidoCartas=false;
		for ( Iterator<Carta> it = list.iterator();it.hasNext();){

			Carta carta =(Carta)it.next();
			MiSprite sprt=carta.getSprite();

			if (carta.getSituacion()==3&&sprt.getBocaAbajo()){

				movidoCartas=true;
				if(!it.hasNext()){
					sprt.registerEntityModifier(new
MoverX(0.3f,sprt.getX(),452){
						protectedvoid
onModifierFinished(IEntity pItem) {
							recogerMazo3();

							super.onModifierFinished(pItem);
						}
					});
				}else{
					sprt.registerEntityModifier(new
MoverX(0.3f,sprt.getX(),452));
				}

			}
			System.out.println("abajo amontonamos"+carta.getNombre()+"
que tiene siguente "+it.hasNext());
		}
		if(!movidoCartas)recogerMazo3();

	}


	publicvoid reparteCartasZ(int numJugadores){
		this.numJugadores=numJugadores;
		Baraja baraja=game.getBaraja();
		intz=0;
			Iterator<Carta> it = baraja.getList().iterator();
			try {
				Thread.currentThread().sleep(1000);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			Carta carta = (Carta)it.next();
			MiSprite sprt =carta.getSprite();
			sprt.setPosition(0, 0);
			sprt.translationZ(-1500);
			sprt.verDorso();
			sprt.setVisible(true);
			intpos=carta.getNumJugador();
```

```
                    try {
                          Thread.currentThread().sleep(1000);
                    } catch (InterruptedException e1) {
                          // TODO Auto-generated catch block
                          e1.printStackTrace();
                    }
                    //sprt.registerEntityModifier(new
MoveModifier(1f,0,0,100,100));
                    sprt.registerEntityModifier(new
MoverX(1f,sprt.getX(),1700));
                    //sprt.registerEntityModifier(new
MoveYModifier(1f,sprt.getY(),-1500));
                    try {
                          Thread.currentThread().sleep(1000);
                    } catch (InterruptedException e1) {
                          // TODO Auto-generated catch block
                          e1.printStackTrace();
                    }
                    //sprt.registerEntityModifier(new MoveZModifier(10f,0,-
1500));
for(;;){
      //System.out.println("laposiciones X="+sprt.getX());
      try {
            Thread.currentThread().sleep(300);
      } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }

}

      }



      publicint[] posJugador(int pos) {

            int[] retorno = { 0, 0 };
            switch (pos) {

            case 1:
                  retorno = PosJ1;
                  break;
            case 2:
                  retorno = PosJ2;
                  break;
            case 3:
                  retorno = PosJ3;
                  break;
            case 4:
                  retorno = PosJ4;
                  break;
            }
            return retorno;
      }
}
```

Archivo DirectorJuego.java

```
package xnetcom.pro.cartas.juego;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.concurrent.CountDownLatch;

import org.anddev.andengine.entity.IEntity;

import org.anddev.andengine.entity.scene.CameraScene;

import org.anddev.andengine.entity.text.ChangeableText;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.jugadores.Jugador;
import xnetcom.pro.cartas.jugadores.JugadorBot;
import xnetcom.pro.cartas.jugadores.JugadorHumano;
import xnetcom.pro.cartas.red.DirectorOnline;
import xnetcom.pro.cartas.sprites.BlogSprite;
import xnetcom.pro.cartas.sprites.Dado;
import xnetcom.pro.cartas.sprites.MiSprite;
import xnetcom.pro.cartas.sprites.MoverX;
import xnetcom.pro.cartas.sprites.MoverY;

publicclass DirectorJuego {
	intnumJugadores;
	Baraja baraja;
	Jugador[] jugadores;
	JugadorHumano humano;
	intturno;
	private Dado dado;
	private CameraScene scene;
	private Thread thread;
	private Game game;
	private CountDownLatch CD;
	ChangeableText puntJ1;
	ChangeableText puntJ2;
	ChangeableText puntJ3;
	ChangeableText puntJ4;
	private Carta CartaJuego;
	BlogSprite blog;
	privateintnumerojugadas=46;
	private HashSet<Integer>hs;
	publicbooleanonline;
	privateinttipos[];
	private DirectorOnline DO;

	public DirectorJuego(Game game, Dado dado, CameraScene scene) {
		this.game = game;
		this.scene = scene;
		this.dado = dado;
		thread = new HiloJuego();
		hs=new HashSet<Integer>();
	}

	public CountDownLatch getCD() {
```

```
		returnCD;
	}

	public Jugador[] getJugadores() {
		returnjugadores;
	}
	publicvoid setJugadores(Jugador[] jugadores){
		this.jugadores =jugadores;
	}

	public Carta getCartaJuego() {
		returnCartaJuego;
	}

	publicint getNumJugadores() {
		returnnumJugadores;
	}

	public String NombreAleatorio() {

		Random generator = new Random();
		int aStart = 1;
		int aEnd = 10;
		int aleatorio;
		do{
			long range = (long) aEnd - (long) aStart + 1;
			long fraction = (long) (range * generator.nextDouble());
			aleatorio = (int) (fraction + aStart);
		}while(!hs.add(aleatorio));

		switch (aleatorio) {

		case 1:
			return"Miguel";
		case 2:
			return"Angel";
		case 3:
			return"Pedro";
		case 4:
			return"Ana";
		case 5:
			return"Merche";
		case 6:
			return"Carlos";
		case 7:
			return"Natxo";
		case 8:
			return"Maria";
		case 9:
			return"Wilson";
		case 10:
			return"Carlota";
		}
		return"usuario";
	}

	publicvoid nuevaPartida(int numJugadores,boolean online, Baraja
baraja, int... tipos) {
		//online=true;//kietarestoo
```

```
            //this.online=online;
            this.tipos=tipos;

            puntJ1 = new ChangeableText(600, 440, game.getFont(), "0 pts",
                        "99999 pts".length());
            puntJ2 = new ChangeableText(50, 6, game.getFont(), "0 pts",
                        "99999 pts".length());
            puntJ3 = new ChangeableText(360, 6, game.getFont(), "0 pts",
                        "99999 pts".length());
            puntJ4 = new ChangeableText(700, 6, game.getFont(), "0 pts",
                        "99999 pts".length());
            scene.attachChild(puntJ1);
            scene.attachChild(puntJ2);
            scene.attachChild(puntJ3);
            scene.attachChild(puntJ4);

            this.numJugadores = numJugadores;
            this.baraja = baraja;
            blog = game.getblog();

            jugadores = new Jugador[numJugadores];
            jugadores[0] = new JugadorHumano(game, Jugador.HUMANO, 1,
"Jugador");
            humano = (JugadorHumano) jugadores[0];

            if(online){

                  DO= new DirectorOnline(game, this);
                  game.setDirectorOnline(DO);
                  if (DO.NuevaPartida()==-1){
                        //poneralgosi ha idomal
                        game.tostar("error al de conexion");
                  }else jugar();
            }
            else{  // si no es online

                  for (int i = 1; i < numJugadores; i++) {
                        jugadores[i] = new JugadorBot(game, Jugador.BOT, i
+ 1, NombreAleatorio());
                  }


                  baraja.reparte(numJugadores);

                                    ArrayList<Carta> list =
baraja.getList();
                  for (Iterator<Carta> it = list.iterator(); it.hasNext();)
{
                        Carta carta = (Carta) it.next();
                        if (carta.getNumJugador() != 0)
      jugadores[carta.getNumJugador() - 1].addCarta(carta);
                        System.out.println("carta
"+carta.getNombre()+"numero"+carta.getNumero());
                        carta.setJugador(jugadores[carta.getNumJugador() -
1]);
                        if (carta.getNumJugador() == 1)
                              carta.setSituacion(2);
                  }
```

```
                jugar();
            }

        }


        publicvoid verCartas(Baraja baraja){
            this.baraja = baraja;
            jugadores = new Jugador[1];
            jugadores[0] = new JugadorHumano(game, Jugador.HUMANO, 1,
"Jugador");
            humano = (JugadorHumano) jugadores[0];
            baraja.reparteVerCartas();

            ArrayList<Carta> list = baraja.getList();
            for (Iterator<Carta> it = list.iterator(); it.hasNext();) {
                Carta carta = (Carta) it.next();
                if (carta.getNumJugador() != 0)
        jugadores[carta.getNumJugador() - 1].addCarta(carta);
                System.out.println("carta
"+carta.getNombre()+"numero"+carta.getNumero());
                carta.setJugador(jugadores[carta.getNumJugador() - 1]);
                if (carta.getNumJugador() == 1)
                    carta.setSituacion(2);
            }

            game.getDirector3D().reparteCartas(numJugadores);
        }

        publicvoid jugar() {
            thread.start();
        }

        public JugadorHumano gethumano() {
            returnhumano;
        }

        privateclass HiloJuego extends Thread {

            publicvoid run() {
                // sleep 1000 porsi el sistemaeslentoqueledetiempo a carar
                // bien el juego


                // sies online el juegotodavia no
tienerepartidaslascartasnisabecuantosjugadores hay
                if(online){

                    // esperar a quenosinicielapartida
                    DO.getSincro().EsperaPartida();

        numJugadores=DO.getDatosTCP_ini().getNumJugadores();
                    //baraja.reparte(numJugadores);
                    humano = (JugadorHumano) jugadores[0];
                    switch (numJugadores) {
                    case 2:
                        baraja.reparteOnline(numJugadores,
DO.getDatosJugadores()[0].getBaraja(),DO.getDatosJugadores()[1].getBaraja());
                        break;
```

```
                        case 3:
                              baraja.reparteOnline(numJugadores,
DO.getDatosJugadores()[0].getBaraja(),DO.getDatosJugadores()[1].getBaraja(),D
O.getDatosJugadores()[2].getBaraja());
                              break;
                        case 4:
                              baraja.reparteOnline(numJugadores,
DO.getDatosJugadores()[0].getBaraja(),DO.getDatosJugadores()[1].getBaraja(),D
O.getDatosJugadores()[2].getBaraja(),DO.getDatosJugadores()[3].getBaraja());
                              break;
                        }
                                                ArrayList<Carta> list =
baraja.getList();
                        for (Iterator<Carta> it = list.iterator();
it.hasNext();) {
                              Carta carta = (Carta) it.next();
                              if (carta.getNumJugador() != 0)
      jugadores[carta.getNumJugador() - 1].addCarta(carta);

      carta.setJugador(jugadores[carta.getNumJugador() - 1]);
                              if (carta.getNumJugador() == 1)
                                    carta.setSituacion(2);
                        }

      game.getDirector3D().reparteCartas(numJugadores);

                        DO.enviarCLI_PARTIDA_INI_ACK();
                        DO.getSincro().EsperaDados();
                        String tirada =DO.getDatosTCP().getTirada();
                        if (jugadores[0].getNombre().equals(tirada))
dado.tirarDadoConResultado(1);
                        elseif (jugadores[0].getNombre().equals(tirada)){
                              dado.tirarDadoConResultado(1);
                              turno = 0;
                        }
                        elseif (jugadores[1].getNombre().equals(tirada)){
                              dado.tirarDadoConResultado(2);
                              turno = 1;
                        }
                        elseif (jugadores[2].getNombre().equals(tirada)){
                              dado.tirarDadoConResultado(3);
                              turno = 2;
                        }
                        elseif (jugadores[3].getNombre().equals(tirada)){
                              dado.tirarDadoConResultado(4);
                              turno = 3;
                        }

                  }else{

                        try {
                              sleep(1000);
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }

                        // baraja.medias();
                        game.getDirector3D().reparteCartas(numJugadores);
```

```
                        dado.tirarDado(numJugadores);
                        System.out.println("la tirada es " +
dado.getTirada());
                        turno = dado.getTirada() - 1;
                        // turno=0;
                  }

                  for (int h=0;h<numerojugadas;h++) {
                        CD = new CountDownLatch(1);
                        System.out.println("llamando turno del jugador " +
(turno + 1));
                        jugadores[turno].turno(true, null);
                        System.out.println("llamado turno");

                        try {
                              System.out.println("espera turno");
                              CD.await();
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                        System.out.println("salido espera");

                        CartaJuego =
jugadores[turno].getCartaSeleccionada();
                        // carta.getSprite().setZIndex(100);
                        // if(carta.getNumJugador()!=1)System.out.println("
j");
                        // //game.getDirector3D().CartaAretar(carta);
                        // else
game.getDirector3D().CartaAretarHumano(carta);

                        CD = new CountDownLatch(jugadores.length - 1);//
tenerencuanta

                        // luego

                        // jugadores

                        // eliminados

                        for (int i = 0; i <jugadores.length; i++) {
                              if (i != turno)
                                    jugadores[i].turno(false, CartaJuego);
                        }

                        try {
                              CD.await();
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                        System.out.println("salido espera de los 3");
                        // esperar a quelos 4 hagansuselecciones

                        for (int i = 0; i <jugadores.length; i++) {

      jugadores[i].getCartaSeleccionada().getSprite()
                                          .verCarta(0.5f);
```

```
                        }

                        try {
                              sleep(2000);
                        } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }

                        Carta cartas[] = new Carta[10];
                        intpuntos=0;
                        // algoritmoparadeterminarquienganalamano
                        float valor = 0;
                        int posicion = -1;
                        intganador;
                        for (int i = 0; i <jugadores.length; i++) {
                              cartas[i] =
jugadores[i].getCartaSeleccionada();
                              int seleccion =
jugadores[i].getCartaSeleccionada()
                                          .getSeleccion();

                              if (valor <=
jugadores[i].getCartaSeleccionada()
                                          .getValorSeleccion(seleccion)) {
                                    valor =
jugadores[i].getCartaSeleccionada()

      .getValorSeleccion(seleccion);
                                    posicion = i;
                                    System.out.println(" jugador "
                                                + (posicion + 1)
                                                + " tiene una carta valor
"
                                                +
jugadores[i].getCartaSeleccionada()

      .getValorSeleccion(seleccion));
                                    System.out.println("el maximo es " +
valor);
                              }
                        }
                        if(online){
                              DO.getSincro().Espera_Fin_Ronda();
                              puntos=DO.getDatosTCP().getPuntos();

      posicion=DO.buscaJugadorPorUsuario(DO.getDatosTCP().getGanador()).getN
umJugador();
                              posicion--;
                              blog.ponerDatos(cartas[0], cartas[1],
cartas[2], cartas[3],     posicion +1);

                        }else{
                              blog.ponerDatos(cartas[0], cartas[1],
cartas[2], cartas[3],     posicion + 1);
                        }


                        CD = new CountDownLatch(1);
```

```
                        try {
                              CD.await();
                        } catch (InterruptedException e1) {
                              // TODO Auto-generated catch block
                              e1.printStackTrace();
                        }

                        for (int i = 0; i <jugadores.length; i++) {
                              if (i == posicion) {

      jugadores[i].GanarCartas(jugadores.length - 1);
                                    jugadores[i].sumaPuntos(188 -
jugadores[i]

      .getCartaSeleccionada().getPidh());
                                    System.out.println("gano el jugador "
+ (posicion + 1)
                                                + " y gana " +
(jugadores.length - 1)
                                                + " cartas");

                                    MiSprite sprt =
jugadores[i].getCartaSeleccionada()
                                                .getSprite();
                                    jugadores[i].getBaraja().getList()

      .remove(jugadores[i].getCartaSeleccionada());

                                    sprt.registerEntityModifier(new
MoverX(0.4f, sprt
                                                .getX(),
game.getDirector3D().posJugador(
                                                posicion + 1)[0]));
                                    sprt.registerEntityModifier(new
MoverY(0.4f, sprt
                                                .getY(),
game.getDirector3D().posJugador(
                                                posicion + 1)[1]) {
                                          protectedvoid
onModifierFinished(IEntity pItem) {
                                                MiSprite df = (MiSprite)
pItem;
                                                df.setVisible(false);

      super.onModifierFinished(pItem);
                                                df.remover(df);
                                          }
                                    });
                                    sprt.verDorso(0.2f);

                              } else {
                                    MiSprite sprt =
jugadores[i].getCartaSeleccionada()
                                                .getSprite();
                                    jugadores[posicion].sumaPuntos(188 -
jugadores[i]

      .getCartaSeleccionada().getPidh());
```

```
                                        //
jugadores[i].getCartaSeleccionada().getSprite().
                                        jugadores[i].getBaraja().getList()

      .remove(jugadores[i].getCartaSeleccionada());
                                        sprt.enpequenecer(0.2f);
                                        sprt.registerEntityModifier(new
MoverX(0.4f, sprt
                                                .getX(),
game.getDirector3D().posJugador(
                                                posicion + 1)[0]));
                                        sprt.registerEntityModifier(new
MoverY(0.4f, sprt
                                                .getY(),
game.getDirector3D().posJugador(
                                                posicion + 1)[1]) {
                                            protectedvoid
onModifierFinished(IEntity pItem) {
                                                MiSprite df = (MiSprite)
pItem;
                                                df.setVisible(false);

      super.onModifierFinished(pItem);
                                                df.remover(df);

                                            }
                                        });
                                        sprt.verDorso(0.2f);
                                        System.out.println("el jugador " + (i
+ 1)
                                                + " perdio su carta");
                                }

                        }
                        game.getScene().sortChildren();
                        for (int i = 0; i <jugadores.length; i++) {
                                if (i == 0)
                                        puntJ1.setText(jugadores[i].puntos() +
" pts");
                                if (i == 1)
                                        puntJ2.setText(jugadores[i].puntos() +
" pts");
                                ;
                                if (i == 2)
                                        puntJ3.setText(jugadores[i].puntos() +
" pts");
                                ;
                                if (i == 3)
                                        puntJ4.setText(jugadores[i].puntos() +
" pts");
                                ;
                        }
                        turno = posicion;

                        // turno=0;
                        try {
                                sleep(1000);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
```

```
                                e.printStackTrace();
                        }


      System.out.println("game.getScene().getChildCount()"+
game.getScene().getChildCount());
                  }// forinfinito

                  intganador=0;
                  int puntuacionMax=0;
                  for (int i = 0; i <jugadores.length; i++) {
                        if (puntuacionMax<=jugadores[i].puntos()){
                              puntuacionMax=jugadores[i].puntos();
                              ganador=i;
                        }
                  }
                  ganador++;

                  CD = new CountDownLatch(1);
                  blog.setBlogFinal(jugadores);
                  blog.setVisible(true);
                  try {
                        System.out.println("espera turno");
                        CD.await();
                  } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
                  //akiterminarialapartidaa
                  game.finish();

            }// run
      }// class hilo

}// class
```

Archivo DirectorTactil.java

```
package xnetcom.pro.cartas.juego;

import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.Scene.IOnSceneTouchListener;
import org.anddev.andengine.input.touch.TouchEvent;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.sprites.MiSprite;

publicclass DirectorTactil implements IOnSceneTouchListener{

	private Game juego;
	MiSprite msprt=null;
	privateintoldX=0;
	intmovidoX=0;
	intmovidoY=0;
	privateintoldY;
	privatelongtiempo;
	booleanmovido=false;

	booleanpillado=false;
	intnumero=-1;



	public DirectorTactil(Game juego){
		super();
		this.juego=juego;
	}

	@Override
	publicboolean onSceneTouchEvent(Scene pScene, TouchEvent
pSceneTouchEvent) {
		if (!pillado)seleccionCarta(pSceneTouchEvent);
		if (msprt!=null){
			if(msprt.enMarcha()){

	System.out.println("msprt.enMarcha()"+msprt.enMarcha()+"la
acarta"+msprt.getNombre());
				returntrue;
			}
			if(msprt.getCarta().getSituacion()==2){

					juego.getDirector3D().disponerMazo();

			}else {

				msprt.onAreaTouched(pSceneTouchEvent, 0, 0);
			}
		}
		if(pSceneTouchEvent.getAction()==1){
			msprt=null;
			pillado=false;
			returnfalse;
		}
```

```
			returntrue;
		}


		publicboolean click(){
			if(System.currentTimeMillis()-tiempo<=300&&!movido)returntrue;
			elsereturnfalse;
		}

		//funcionquecorrigeunfalloen AndEngine
alseleccionarspritesqueseencuentranenlamismaposicion
		publicvoid seleccionCarta(TouchEvent pSceneTouchEvent){
			if(juego.getTic().contains(pSceneTouchEvent.getX(),
pSceneTouchEvent.getY())&&juego.getTic().isVisible()){
				juego.getTic().onAreaTouched(pSceneTouchEvent, 0,0);
				return;
			}
			if(juego.getblog().contains(pSceneTouchEvent.getX(),
pSceneTouchEvent.getY())&&juego.getblog().isVisible()){
				juego.getblog().onAreaTouched(pSceneTouchEvent, 0,0);
				return;
			}

			for(int i=0;i<juego.getSprites().size();i++){

		if(juego.getSprites().get(i).contains(pSceneTouchEvent.getX(),
pSceneTouchEvent.getY())&&juego.getSprites().get(i).isVisible()){
					if(msprt==null)	msprt=juego.getSprites().get(i);
					else{

		if(msprt.getZIndex()<juego.getSprites().get(i).getZIndex()){

							msprt=juego.getSprites().get(i);
					}

					}
				}
			}
		}


		publicboolean pillar(int numero){
			if(!pillado||this.numero==numero){
				pillado=true;
				this.numero=numero;
				returntrue;
			}
			elsereturnfalse;
		}
		publicvoid soltar(){
			pillado=false;
		}


}
```

Archivo Jugador.java

```
package xnetcom.pro.cartas.jugadores;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Baraja;
import xnetcom.pro.cartas.juego.Carta;

publicabstractclass Jugador {
	publicstaticfinalintHUMANO =1;
	publicstaticfinalintBOT =2;
	publicstaticfinalintREMOTO =3;

	protectedinttipoJugador;
	protectedintNumeroJugador;
	protected  Baraja baraja;
	protectedbooleaneliminado=false;
	protectedbooleansoyMano=false;
	protected Carta cartaSalida;
	protected Carta cartaEntrada;
	protectedintcartasganadas=0;
	protected Game game;
	protectedbooleansacarCarta;
	protectedbooleanturno;
	protectedintnumCartas3d=0;
	protectedbooleanCartavista=false;
	protectedintpuntuacion=0;
	protected String nombre;
	public Jugador(Game game,int tipoJugador,int NumeroJugador, String
nombre ){
		/*
		 * tipoJugador 1 Humano
		 * tipoJugador 2 Bot
		 * tipoJugador 3 Remoto
		 */
		this.nombre=nombre;
		this.tipoJugador=tipoJugador;
		this.NumeroJugador=NumeroJugador;
		baraja =new Baraja();
		this.game=game;

	}

	public String getNombre() {
		returnnombre;
	}
	publicint getNumJugador(){
		returnNumeroJugador;
	}
	publicvoid sumaPuntos(int puntos){
		puntuacion+=puntos;
	}
	publicint puntos(){
		returnpuntuacion;
	}
	publicboolean CartaVista(){
		returnCartavista;
	}
	publicvoid setCartaVista(boolean boleana){
```

```
		Cartavista=boleana;
	}
	publicint getTipojugador() {
		returntipoJugador;
	}
	publicsynchronizedvoid addCartas3d(){
		numCartas3d++;
	}
	publicsynchronizedint getCartas3d(){
		returnnumCartas3d;
	}

	publicboolean turno() {
		returnturno;
	}
	publicboolean sacarCarta(){
		returnsacarCarta;
	}
	publicvoid addCarta(Carta carta) {
	baraja.add(carta);
	}
	publicint  CartasGanadas(){
		returncartasganadas;
	}
	publicvoid  GanarCartas(int ganadas){
		cartasganadas+=ganadas;
	}

	public Baraja getBaraja(){
		returnbaraja;
	}
	publicint getNumeroCartas(){
		returnbaraja.getList().size();
	}
	publicboolean getElminado(){
		returneliminado;
	}
	publicvoid setElminado(){
		eliminado=true;
	}
	publicint getNumCartas(){
		returnbaraja.getList().size();
	}

	publicvoid turno(boolean sacarCarta,Carta card){
	}

	public Carta getCartaSeleccionada(){
		returncartaSalida;
	}

}
```

Archivo JugadorBot,java

```
package xnetcom.pro.cartas.jugadores;

import java.util.Random;
import java.util.Collections;
import java.util.Iterator;


import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Carta;
import xnetcom.pro.cartas.sprites.MiSprite;

publicclass JugadorBot extends Jugador {

	private Random generator;
	private HiloJugador hiloJugador;


	public JugadorBot(Game game, int tipoJugador, int NumeroJugador,
String nombre) {
		super(game, tipoJugador, NumeroJugador, nombre);
		generator = new Random();
		hiloJugador=new HiloJugador();
		// TODO Auto-generated constructor stub
	}


	// true, tetocasacarcarta, false ya ha sacadocarta y esesa
	publicvoid turno(boolean sacarCartaAretar, Carta card) {

		hiloJugador=new HiloJugador();

		sacarCarta=sacarCartaAretar;
		cartaEntrada=card;

		hiloJugador.start();


	}

	privatebooleantomaDecision(int porcentaje) {
		int numero = tomaDecision(1, 100);
		if (porcentaje >= numero) {
			returntrue;
		} else
			returnfalse;
	}

	privateint tomaDecision(int aStart, int aEnd) {

		long range = (long) aEnd - (long) aStart + 1;
		long fraction = (long) (range * generator.nextDouble());
		int aleatorio = (int) (fraction + aStart);
		return aleatorio;

	}
```

```
	privateclass HiloJugador extends Thread{


		public Float ajustaIDH(Float valor){
			if(valor>=0.9f) return valor*1.2f;
			if(valor>=0.85) return valor*1.1f;
			if(valor<=0.4)return valor*0.9f;
			return valor;
		}
		public Float ajustaEDU(Float valor){
			if(valor>=11)return valor*1.5f;
			if(valor>=9)return valor*1.2f;
			if(valor<=7)return valor*0.9f;
			return valor;
		}
		public Float ajustaPIB(Float valor){
			if(valor>=40000) return valor*1.5f;
			if(valor>=35500) return valor*1.3f;
			if(valor<=1000) return valor*0.8f;
			return valor;
		}
		public Float ajustaSALUD(Float valor){
			if(valor>=81) return valor*1.2f;
			if(valor>=80) return valor*1.1f;
			if(valor<=51) return valor*0.8f;

			return valor;
		}

		publicvoid inteligencia2(){

			int numerocarta = tomaDecision(0, getNumeroCartas() - 1);
			cartaSalida= getBaraja().getList().get(numerocarta);
			MiSprite sprt=cartaSalida.getSprite();

			int
valores[]=game.getBaraja().posiciones(cartaSalida.getSalud(),
cartaSalida.getEducacion(), cartaSalida.getIngresos());
			int pid=cartaSalida.getPidh();
			int valor=pid;
			sprt.setSeleccionBot(1);
			cartaSalida.setSeleccion(1);
			System.out.println("pospid "+pid);


			//unpocodealeatoridad
			valores[0]=tomaDecision(valores[0]-10, valores[0]+10);//
posicionporsalud
			valores[1]=tomaDecision(valores[1]-10, valores[1]+10);//
posicionporeducacion
			valores[2]=tomaDecision(valores[2]-10, valores[2]+10);//
posicionporpib
			System.out.println("aleatorio possalud "+valores[0]+"edu
"+valores[1]+" pib "+valores[2]);

			if(valor>valores[0]){
				sprt.setSeleccionBot(2);
```

```
                        cartaSalida.setSeleccion(2);
                        valor=valores[0];
                  }
                  if(valor>valores[1]){
                        sprt.setSeleccionBot(3);
                        cartaSalida.setSeleccion(3);
                        valor=valores[1];
                  }
                  if(valor>valores[2]){
                        sprt.setSeleccionBot(4);
                        cartaSalida.setSeleccion(4);
                        valor=valores[2];
                  }
            }
            publicvoidinteligencia1(){

                  int numerocarta = tomaDecision(0, getNumeroCartas() - 1);
                  cartaSalida= getBaraja().getList().get(numerocarta);
                  MiSprite sprt=cartaSalida.getSprite();


                  Float idh=cartaSalida.getIdh();
                  Float salud=cartaSalida.getSalud();
                  Float educacion=cartaSalida.getEducacion();
                  Float ingresos=cartaSalida.getIngresos();


                  //condicionesdeajustedesupuestainteligencia

                  idh= ajustaIDH(idh);
                  educacion=ajustaEDU(educacion);
                  ingresos=ajustaPIB(ingresos);
                  salud=ajustaSALUD(salud);

                  //correcionesparacompararporiguallosvalores
                  idh=idh*1.07f;
                  salud=salud*(0.010627f);
                  ingresos=ingresos*(0.00002182f);
                  educacion=educacion*(0.07784f);

                  //funcionaleatoriaparaquehayamasvariedaddeelecciones no
tantoidh y salud

                  ingresos=ingresos*(1+(tomaDecision(1, 30))/100);
                  educacion=educacion*(1+(tomaDecision(1, 30))/100);

                  // tomala decision deque valor tomacogiendo el mayor
                  float valorMAX =idh;

                  sprt.setSeleccionBot(1);
                  cartaSalida.setSeleccion(1);
                  System.out.println("idh real=" +cartaSalida.getIdh());
                  System.out.println("idh correccion=" +idh);
                  System.out.println("idh posicion="
+cartaSalida.getPidh());

                  System.out.println("salud real=" +cartaSalida.getSalud());
                  System.out.println("valor de salud ="+salud);
```

```
                    System.out.println("edu real="
+cartaSalida.getEducacion());
                    System.out.println("valor de edu ="+educacion);
                    System.out.println(" ingresos real ="
+cartaSalida.getIngresos());
                    System.out.println("valor de ingresos ="+ingresos);


                    if (valorMAX < salud) {
                          valorMAX = salud;
                          sprt.setSeleccionBot(2);
                          cartaSalida.setSeleccion(2);

                    }

                    if (valorMAX < educacion) {
                          valorMAX = educacion;
                          sprt.setSeleccionBot(3);
                          cartaSalida.setSeleccion(3);

                    }

                    if (valorMAX < ingresos) {
                          valorMAX = ingresos;
                          sprt.setSeleccionBot(4);
                          cartaSalida.setSeleccion(4);

                    }
                    //sprt.setSeleccion(2);
                    System.out.println("seleccionamos el valor " +
cartaSalida.getSeleccion());
                    System.out.println("nombre"+cartaSalida.getNombre());


             }


      intRango=1; //rangodedestreza a mayor rangomenordestrezabotminimo 0

             publicvoid run(){


                    if (sacarCarta) {

                          //inteligencia1();
                          inteligencia2();

                    }
                    if (!sacarCarta) {
                          Carta carta=null;
                          //System.out.println("segundojugadorrr");
                          int seleccion = cartaEntrada.getSeleccion();
                          float valor =
cartaEntrada.getValorSeleccion(seleccion);

                          for (Iterator<Carta> it =
getBaraja().getList().iterator(); it.hasNext();) {
                                carta = (Carta) it.next();// ojoconmachacarlo
                                carta.setSeleccion(seleccion);
```

```
                        }

                        Collections.sort(getBaraja().getList());
                        int i = 0;
                        int encontrado = -1;
                        for (Iterator<Carta> it =
getBaraja().getList().iterator(); it.hasNext() && encontrado == -1; i++) {
                              carta = (Carta) it.next();
                              if (valor <
carta.getValorSeleccion(seleccion)) {
                                    encontrado = i;
                              }
                        }
                        if (encontrado != -1) {
                              boolean salir=true;
                              do {
                                    int eleccion = tomaDecision(encontrado
- Rango, encontrado + Rango);

                                    try {

      //System.out.println("tomadecisionconvaloresentre"+(encontrado - 1) +"
y "+(encontrado + 1));
                                          System.out.println("el elegido
es "+encontrado +" seleccionamo "+ eleccion+" la carta elegida es
"+getBaraja().getList().get(eleccion).getNombre());
                                          carta =
getBaraja().getList().get(eleccion);
                                          salir=true;
                                    } catch (Exception e) {
                                          System.out.println("excepcion el
sacar algo fuera de lista");
                                          salir = false;
                                    }
                              } while (!salir);

                        } else {
                              //si no cogeunacualquiera
                              int eleccion = tomaDecision(0,
baraja.getList().size()-1);
                              carta = baraja.getList().get(eleccion);

                        }
                        cartaSalida=carta;
                  }

                  try {
                        sleep(tomaDecision(1000, 3000));
                  } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
                  //sprt.setZIndex(50);
                  intpos=baraja.getPosicion(cartaSalida);
                  boolean encontrado=false;
                  if (cartaSalida.getSprite().isVisible()) {
                        for (Iterator<Carta> it =
baraja.getList().iterator(); it.hasNext() &&!encontrado;) {
                              MiSprite naipe = it.next().getSprite();
```

```
                                if (!naipe.isVisible()) {
                                      naipe.setVisible(true);
                                      encontrado = true;
                                }
                          }
                    }
                    cartaSalida.getSprite().censura();

      if(!sacarCarta)game.getDirector3D().sacarCartaBocaAbajo(cartaSalida);
                    elsegame.getDirector3D().sacarCarta(cartaSalida);

      System.out.println("game.getDirectorJuego().getCD().countDown();");
                    game.getDirectorJuego().getCD().countDown();
                    //return carta;
                    return;
              }


      }

}
```

Archivo jugadorHumano.java

```
package xnetcom.pro.cartas.jugadores;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Carta;

publicclass JugadorHumano extends Jugador{
privatebooleanticListo=false;

	intsel=0;
	public JugadorHumano(Game game, int tipoJugador, int NumeroJugador,
String nombre) {
		super(game, tipoJugador, NumeroJugador, nombre);
		// TODO Auto-generated constructor stub
	}
	publicvoid turno(boolean sacarCarta,Carta card){
		this.sacarCarta=sacarCarta;
		cartaEntrada=card;
		if(!sacarCarta)sel=card.getSeleccion();
		turno=true;
	}
	publicvoid turno(boolean sacarCarta){
		this.sacarCarta=sacarCarta;
		turno=true;
	}

	publicboolean ticListo(){
		returnticListo;
	}
	publicint getSel(){
		returnsel;
	}

	publicvoid setTicListo(boolean ticListo){
		this.ticListo=ticListo;
	}
	publicvoid cartaSeleccionada(Carta carta){
		if(game.isOnline()){
			if(sacarCarta){

	game.getDirectorOnline().enviarCLI_PARTIDA_MANO_INI(this.getNombre(),S
tring.valueOf(carta.getNumero()),String.valueOf(carta.getSeleccion())) ;
			}else{

	game.getDirectorOnline().enviarCLI_PARTIDA_JUGADA(this.getNombre(),Str
ing.valueOf(carta.getNumero()),String.valueOf(carta.getSeleccion())) ;
			}
		}
		cartaSalida=carta;
		cartaSalida.setSituacion(6);
		game.getDirector3D().CartaAretarHumano(carta);
		game.getDirectorJuego().getCD().countDown();
		sacarCarta=false;
		turno=false;
		Cartavista=false;
	}
}
```

Archivo JugadorRemoto.java

```
package xnetcom.pro.cartas.jugadores;


import java.util.Iterator;


import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Carta;
import xnetcom.pro.cartas.sprites.MiSprite;




publicclass JugadorRemoto extends Jugador{


	private HiloJugador hiloJugador;
	private datosSincronizados datos;
	private accionSincro sincro;

	public JugadorRemoto(Game game, int tipoJugador, int NumeroJugador,
String nombre) {
		super(game, tipoJugador, NumeroJugador, nombre);
		sincro=new accionSincro();
		hiloJugador=new HiloJugador();
		datos=new datosSincronizados();
		// TODO Auto-generated constructor stub
	}

	publicvoid turno(boolean sacarCartaAretar, Carta card) {

		hiloJugador=new HiloJugador();

		sacarCarta=sacarCartaAretar;
		cartaEntrada=card;

		hiloJugador.start();

	}

	public datosSincronizados GetDatosSincronizados() {
		returndatos;
	}

	privateclass HiloJugador extends Thread{


		publicvoid SacarCartaOnline(){

			// aquiesperar a que el jugador online sacaraunacarta
			int[] accion=sincro.EsperaAccion();

			//int[]salida=datos.getCartaOnline();
			System.out.println("soy jugador online saco carta y
selceeion"+accion[0]+" "+accion[1]);
```

```
                    cartaSalida= getBaraja().getCarta(accion[0]);
                    cartaSalida.setSeleccion(accion[1]);
                    MiSprite sprt=cartaSalida.getSprite();
                    sprt.setSeleccionBot(accion[1]);

             }

             publicvoid run(){

                    SacarCartaOnline();

                    boolean encontrado=false;
                    if (cartaSalida.getSprite().isVisible()) {
                          for (Iterator<Carta> it =
baraja.getList().iterator(); it.hasNext() &&!encontrado;) {
                                MiSprite naipe = it.next().getSprite();
                                if (!naipe.isVisible()) {
                                      naipe.setVisible(true);
                                      encontrado = true;
                                }
                          }
                    }
                    cartaSalida.getSprite().censura();

      if(!sacarCarta)game.getDirector3D().sacarCartaBocaAbajo(cartaSalida);
                    elsegame.getDirector3D().sacarCarta(cartaSalida);

      System.out.println("game.getDirectorJuego().getCD().countDown();");
                    game.getDirectorJuego().getCD().countDown();
                    return;
             }

      }

      public accionSincro getAccion(){
             returnsincro;

      }
      publicclass accionSincro {

             privatebooleanaccion=false;
             privateintcarta=-1;
             privateintseleccion=-1;


             publicsynchronizedint[] EsperaAccion() {
                    try {
                          while (!accion) {
                                this.wait();
                          }
                    } catch (InterruptedException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
                    accion=false;
                    int temp[]=newint[2];
                    temp[0]=carta;
                    temp[1]=seleccion;
                    carta=-1;
```

```
                  seleccion=-1;
                  return temp;

            }
            publicsynchronizedvoid Accion(int carta, int seleccion) {
                  this.carta=carta;
                  this.seleccion=seleccion;
                  accion=true;
                  this.notify();
            }

      }


      publicclass datosSincronizados {
            privateintnumeroCartaOnline=-1;
            privateintseleccionOnline=-1;

            publicsynchronizedvoid setCartaOnline(int numeroCarta, int
seleccion){
                  numeroCartaOnline=numeroCarta;
                  seleccionOnline=seleccion;
            }
            publicsynchronizedboolean hayDatos(){
                  if(numeroCartaOnline==-1 || seleccionOnline==-1){
                        returnfalse;
                   }
                  returntrue;
            }
      publicsynchronizedint[] getCartaOnline (){
      int [] valores={-1,-1};
      if(numeroCartaOnline!=-1 &&seleccionOnline!=-1){
                    valores[0]=numeroCartaOnline;
                    valores[1]=seleccionOnline;
            numeroCartaOnline=-1;
            seleccionOnline=-1;
              }
      return valores;

          }
      }

}
```

Archivo ClienteTCP.java

```
package xnetcom.pro.cartas.red;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;
import java.net.SocketException;


import javax.xml.parsers.ParserConfigurationException;

import org.anddev.andengine.ui.activity.BaseGameActivity;
import org.xml.sax.SAXException;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.activities.MainMenu;



publicclass clienteTCP {



	publicstatic String SERV_ENCOLADO="SERV_ENCOLADO";
	publicstatic String SERV_DESCONECTAR_ACK ="SERV_DESCONECTAR_ACK";
	publicstatic String SERV_REGISTRAR_OK="SERV_REGISTRAR_OK";
	publicstatic String SERV_LOG_ERROR="SERV_LOG_ERROR";
	publicstatic String SERV_MOSTRAR_PERFIL="SERV_MOSTRAR_PERFIL";
	publicstatic String SERV_STAT="SERV_STAT";
	publicstatic String SERV_PARTIDA_INI="SERV_PARTIDA_INI";
	publicstatic String SERV_PARTIDA_DADOS="SERV_PARTIDA_DADOS";
	publicstatic String SERV_PARTIDA_MANO_INI="SERV_PARTIDA_MANO_INI";
	publicstatic String SERV_PARTIDA_JUGADA="SERV_PARTIDA_JUGADA";
	publicstatic String SERV_PARTIDA_FIN_RONDA="SERV_PARTIDA_FIN_RONDA";
	publicstatic String SERV_PARTIDA_FIN="SERV_PARTIDA_FIN";
	publicstatic String SERV_CHAT="SERV_CHAT";
	publicstatic String SERV_MOSTAR_TEXTO="SERV_MOSTAR_TEXTO";
	publicstatic String SERV_ERROR="SERV_ERROR";
	publicstatic String SERV_MOD_OK="SERV_MOD_OK";


protected DataOutputStream dos;
protected DataInputStream dis;



	private ObjectOutputStream salida;
	public ObjectInputStream entrada;
	private String mensaje = "";
	private Socket cliente;
	//private String servidor = "192.168.2.101";
	private String servidor = "138.100.49.9";
	private MainMenu menu=null;
```

```
	private Game juego=null;
	privatebooleandesconexion=false;
	//private boolean iniciado=false;
	privateintPuerto=1234;
	Parser_TCP parser;
	privateintMensaje_pendiente=0;
	private Datos_TCP datos_tcp;
	private HiloTCP hiloTCP;
	privatebooleannuevosDatos=false;
	private Sincro sincro;
	public clienteTCP(BaseGameActivity contexto, HiloTCP hilo) {
		this.hiloTCP=hilo;
		parser = new Parser_TCP();
		if (contexto instanceof MainMenu){
				menu =(MainMenu) contexto;
			}
			if (contexto instanceof Game){
				juego =(Game) contexto;
			}
		sincro =new Sincro();
	}
	public HiloTCP getHilo(){
		returnhiloTCP;
	}
	publicvoid desconexion(){
		desconexion=true;
		try {
			cliente.shutdownInput();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}



	publicvoid ejecutarCliente() {
		desconexion=false;
		System.out.println("\nejecutarCliente()");
		// conectarsealservidor, obtenerflujos, procesarlaconexión
		try {
			conectarAServidor();	// Paso 1: crearun socket
pararealizarlaconexión
			obtenerFlujos();	// Paso 2:
obtenerlosflujosdeentrada y salida
			procesarConexion();	// Paso 3: procesarlaconexión
		}
		// el servidorcerrólaconexión
		catch (EOFException excepcionEOF) {
			System.err.println("El cliente termino la conexión");
		}

		// procesarlosproblemasquepuedenocurriralcomunicarsecon el
		// servidor
		catch (IOException excepcionES) {
			excepcionES.printStackTrace();
		} finally {
```

```
                    cerrarConexion(); // Paso 4: cerrarlaconexión
             }
      } // fin delmétodo ejecutarCliente

      privatevoid conectarAServidor() throws IOException {
             // crear Socket pararealizarlaconexióncon el servidor
             System.out.println("\nconectarAServidor()"+servidor);
             cliente = new Socket(InetAddress.getByName(servidor), Puerto);
             // mostrarlainformacióndelaconexión
             if(juego!=null) juego.tostar("Conectado a: " +
cliente.getInetAddress().getHostName());
             if(menu!=null) menu.tostar("Conectado a: " +
cliente.getInetAddress().getHostName());
      }

      privatevoid obtenerFlujos() throws IOException {
             // establecerflujodesalidaparalosobjetos
             //salida = new ObjectOutputStream(cliente.getOutputStream());
             dos = new DataOutputStream(cliente.getOutputStream());
             dos.flush();
             //salida.flush(); //
vacíarbúferdesalidaparaenviarinformacióndeencabezado




             // establecerflujodeentradaparalosobjetos
             //entrada = new ObjectInputStream(cliente.getInputStream());
             dis = new DataInputStream(cliente.getInputStream());
             //iniciado=true;
             //System.out.println("iniciado=;"+iniciado);
             sincro.setIniciado();
      }

      privatevoid enviarDatos(String mensaje) {
             // enviarobjetoalservidor
             try {
                    //salida.writeObject(mensaje);
                    System.out.print("envio"+mensaje);
                    dos.writeUTF(mensaje);
                    dos.flush();
                    //salida.flush();
                    //entrada.close();
             }
             // procesarlosproblemasquepuedenocurrir al enviar el objeto
             catch (IOException excepcionES) {
             }
      }

      privatevoidprocesarConexio3n() throws IOException {
             // habilitar campoIntroducir para que el usuario del cliente
pueda enviar mensajes

             do { // procesar mensajes enviados del servidor

                    // leer mensaje y mostrarlo en pantalla
                    try {
                           //mensaje = (String) entrada.readObject();
```

```
                        mensaje = dis.readUTF();
                        if(juego!=null) juego.tostar(mensaje);
                        if(menu!=null) menu.tostar(mensaje);

                        try {
                              datos_tcp = parser.nuevoMensaje(mensaje);
                              if (datos_tcp !=null){
                                    sincro.LlegadoMensaje();

                              }
                        } catch (ParserConfigurationException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        } catch (Exception e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }

                  }


                  catch (Exception e) {
                        //mostrarMensaje("\nSe cerrrooo");
                        System.out.println("Exception e");
                        desconexion();
                  }

            } while (!desconexion);

      } // fin del método procesarConexion




      privatevoid procesarConexion() throws IOException {
            // habilitar campoIntroducir para que el usuario del cliente
pueda enviar mensajes

            do {
                  try {
                        //mensaje = (String) entrada.readObject();
                        mensaje = dis.readUTF();
                        datos_tcp = parser.nuevoMensaje(mensaje);
                        if (datos_tcp !=null){
                              sincro.LlegadoMensaje();
                              desconexion();
                        }
                  }/*

                  catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        desconexion=true;
                  } */
                  catch (ParserConfigurationException e) {
                        // TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                  } catch (SAXException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
                  catch (SocketException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        desconexion=true;
                        menu.tostar("desconectado del servidor");
                        System.out.println("SocketException e");
                        return;
                  }
                  System.out.println("llego al while !desconexxion");
            }while (!desconexion);    // leer mensaje y mostrarlo en
pantalla

      } // fin del método procesarConexion



      // cerrar flujos y socket
      publicvoid cerrarConexion() {
            try {
                  dis.close();
                  dos.close();
                  //salida.close();
                  //entrada.close();
                  cliente.close();
            } catch (IOException excepcionES) {
                  excepcionES.printStackTrace();
                  System.out.println("IOException excepcionES");
            }
            catch (Exception excepcionES) {
                  excepcionES.printStackTrace();
                  //menu.tostar(excepcionES.toString());
                  menu.tostar("Error al conectarse");
                  System.out.println("Exception excepcionESs");
            }
            System.out.println("cerrados todos los sockets");
      }




      publicvoid enviarCLI_MODIFICAR(final String... parametros) {
            System.out.println("enviarCLI_MODIFICAR");
            Thread hilo =new Thread (){
                  publicvoid run(){
                        sincro.EsperaIniciado();
                        StringBuffer output = new StringBuffer();
                        output.append("<?xml version=\"1.0\"
encoding=\"UTF-8\" standalone=\"no\"?>");
                        output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_MODIFICAR</mensaje_ID>");
                        output.append("<datos>");
```

```
                        output.append("<usuario>" + parametros[0]+
"</usuario>");
                        output.append("<password>" + parametros[1]+
"</password>");
                        output.append("<new_password>" + parametros[2]+
"</new_password>");
                        output.append("<nombre>" + parametros[3]+
"</nombre>");
                        output.append("<apellidos>" + parametros[4]+
"</apellidos>");
                        output.append("<pais>" + parametros[5]+ "</pais>");
                        output.append("</datos>");
                        output.append("</mensaje>");
                        enviarDatos(output.toString());
                        System.out.println("esperando respuesta");
                        sincro.EsperaMensaje();


      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_MOD_OK)){


      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());
                              menu.getDatos().setUsuario(parametros[0]);
                              menu.getDatos().setNombre(parametros[3]);
                              menu.getDatos().setApellidos( parametros[4]);
                              menu.getDatos().setPassword(parametros[2]);
                              menu.getDatos().setPais(parametros[5]);
                              menu.getDatos().guardar_config();

      System.out.println("datos_tcp.getDatos()[0]"+datos_tcp.getDatos()[0]);

      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());
                              System.out.println("if (menu!=null)
menu.getMenus().getETusuario().setText(datos_tcp.getDatos()[0]");
                              hiloTCP.desconectar();
                              System.out.println("hiloTCP.desconectar();");
                              //menu.getMenus().cajasDeTexto(false, true);
                              menu.getMenus().actualizaCajas();

                        }

      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_ERROR)){
                              menu.tostar(datos_tcp.getDatos()[0]);
                              hiloTCP.desconectar();
                        }

                  }

            };
            hilo.start();
      }
      publicvoid enviarCLI_MOSTRAR_PERFIL(final String... parametros){
            System.out.println("enviarCLI_MOSTAR_PERFIL");
            Thread hilo =new Thread (){
                  publicvoid run(){
                        sincro.EsperaIniciado();
                        StringBuffer output = new StringBuffer();
                        output.append("<?xml version=\"1.0\"
encoding=\"UTF-8\" standalone=\"no\"?>");
```

```
                        output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_MOSTRAR_PERFIL</mensaje_ID>");
                        output.append("<datos>");
                        output.append("<usuario>"+parametros[0]+
"</usuario>");
                        output.append("<password>"+parametros[1]+
"</password>");
                        output.append("</datos>");
                        output.append("</mensaje>");
                        enviarDatos(output.toString());
                        System.out.println("esperando respuesta");
                        sincro.EsperaMensaje();


      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_MOSTRAR_PERFIL)){


      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());

      menu.getDatos().setUsuario(datos_tcp.getDatos()[0]);

      menu.getDatos().setNombre(datos_tcp.getDatos()[1]);

      menu.getDatos().setApellidos(datos_tcp.getDatos()[2]);

      menu.getDatos().setPais(datos_tcp.getDatos()[3]);
                              menu.getDatos().guardar_config();

      System.out.println("datos_tcp.getDatos()[0]"+datos_tcp.getDatos()[0]);

      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());
                              System.out.println("if (menu!=null)
menu.getMenus().getETusuario().setText(datos_tcp.getDatos()[0]");
                              hiloTCP.desconectar();
                              System.out.println("hiloTCP.desconectar();");
                              //menu.getMenus().cajasDeTexto(false, true);
                              menu.getMenus().actualizaCajas();

                        }

      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_ERROR)){
                              menu.tostar(datos_tcp.getDatos()[0]);
                              hiloTCP.desconectar();
                        }

                  }

            };
            hilo.start();

      }

      publicvoid enviarCLI_REGISTRAR(final String... parametros){
            Thread hilo =new Thread (){
                  publicvoid run(){
                        sincro.EsperaIniciado();
                        StringBuffer output = new StringBuffer();
```

```
                        output.append("<?xml version=\"1.0\"
encoding=\"UTF-8\" standalone=\"no\"?>");
                        output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_REGISTRAR</mensaje_ID>");
                        output.append("<datos>");
                        output.append("<usuario>"+parametros[0]+
"</usuario>");
                        output.append("<password>"+parametros[1]+
"</password>");
                        output.append("<nombre>"+parametros[2]+
"</nombre>");
                        output.append("<apellidos>"+parametros[3]+
"</apellidos>");
                        output.append("<pais>"+parametros[4]+ "</pais>");
                        output.append("</datos>");
                        output.append("</mensaje>");
                        enviarDatos(output.toString());
                        System.out.println("esperando respuesta");
                        sincro.EsperaMensaje();


      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_REGISTRAR_OK)){

      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());
                              menu.getDatos().restaurar();
                              menu.getDatos().setUsuario(parametros[0]);
                              menu.getDatos().setPassword(parametros[1]);
                              menu.getDatos().setNombre(parametros[2]);
                              menu.getDatos().setApellidos(parametros[3]);
                              menu.getDatos().setPais(parametros[4]);
                              menu.getDatos().guardar_config();

      System.out.println("datos_tcp.getDatos()[0]"+datos_tcp.getDatos()[0]);

      System.out.println("menu.getDatos()"+menu.getDatos().getUsuario());
                              System.out.println("if (menu!=null)
menu.getMenus().getETusuario().setText(datos_tcp.getDatos()[0]");
                              hiloTCP.desconectar();
                              System.out.println("hiloTCP.desconectar();");
                              //menu.getMenus().cajasDeTexto(false, true);
                              menu.getMenus().actualizaCajas();

                        }

      if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_ERROR)){
                              menu.tostar(datos_tcp.getDatos()[0]);
                              hiloTCP.desconectar();
                        }
                  }

            };
            hilo.start();

      }


      // parametros usuario, password.
```

```
	publicvoid enviarCLI_JUGAR(String... parametros){

		StringBuffer output = new StringBuffer();
		output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
		output.append("<mensaje protocol_ver=\"1.0\">");
		output.append("<mensaje_ID>CLI_JUGAR</mensaje_ID>");
		output.append("<datos>");
		output.append("<usuario>" + parametros[0]+ "</usuario>");
		output.append("<password>" + parametros[1]+ "</password>");
		output.append("<baraja>" + "1.0"+ "</baraja>");
		output.append("</datos>");
		output.append("</mensaje>");
		enviarDatos(output.toString());

	}

}
```

Archivo DatosTCP.java

```
package xnetcom.pro.cartas.red;

import xnetcom.pro.cartas.auxiliares.DatosJugador;

publicclass Datos_TCP {
	private String tipo;
	private String[] datos;

	privateintnumJugadores;
	privateintturno;
	private String texto;
	privateintcarta;
	privateintseleccion;
	private String jugador;
	private String ganador;
	privateintpuntos;
	private String dados;

	private String Jugador1;
	private String Jugador2;
	private String Jugador3;
	private String Jugador4;

	private String LvLJ1;
	private String LvLJ2;
	private String LvLJ3;
	private String LvLJ4;


	privateintcartaJ1;
	privateintcartaJ2;
	privateintcartaJ3;
	privateintcartaJ4;

	privateintpuntosJ1;
	privateintpuntosJ2;
	privateintpuntosJ3;
	privateintpuntosJ4;

	privateintselJ1;
	privateintselJ2;
	privateintselJ3;
	privateintselJ4;
	private DatosJugador[] datosJugador;

	privateintbarajaJ1[];
	privateintbarajaJ2[];
	privateintbarajaJ3[];
	privateintbarajaJ4[];

	public Datos_TCP(String tipo) {
		this.tipo = tipo;
	}

	public DatosJugador[] getDatosJugadores(){
		returndatosJugador;
	}
```

```
	publicvoid setDatos(String datos[]){
		this.datos=datos;
		if(tipo.equals(Parser_TCP.SERV_PARTIDA_INI)){
			System.out.println("numero de jugadores "+datos[0]);
			numJugadores=Integer.parseInt(datos[0]);
			datosJugador= new DatosJugador[numJugadores];

			String[] arrayCartas;
			//datos j1
			Jugador1=datos[1];
			System.out.println("nombre del jugador1 "+datos[1]);
			datosJugador[0]=new DatosJugador();
			datosJugador[0].setNombre(datos[1]);
			System.out.println("Jugador1"+Jugador1);
			LvLJ1=datos[2];
			arrayCartas = datos[3].split(":");
			barajaJ1= newint[arrayCartas.length];
			for(int i=0;i<arrayCartas.length;i++){
				barajaJ1[i]=Integer.parseInt(arrayCartas[i]);

			}

			datosJugador[0].setBaraja(barajaJ1);
			for(int h=0;h<barajaJ1.length;h++){
				System.out.println("soy datosTCP barajaJ1
H="+h+"barajaJ1 "+barajaJ1[h]);
			}



			//datos j2
			Jugador2=datos[4];
			datosJugador[1]=new DatosJugador();
			datosJugador[1].setNombre(datos[4]);
			LvLJ2=datos[5];
			arrayCartas = datos[6].split(":");
			barajaJ2= newint[arrayCartas.length];
			for(int i=0;i<arrayCartas.length;i++){
				barajaJ2[i]=Integer.parseInt(arrayCartas[i]);

			}
			datosJugador[1].setBaraja(barajaJ2);
			for(int h=0;h<barajaJ2.length;h++){
				System.out.println("soy datosTCP barajaJ2
H="+h+"barajaJ2 "+barajaJ2[h]);
			}


			if(numJugadores<3)return ;
			//datos j3
			Jugador3=datos[7];
			datosJugador[2]=new DatosJugador();
			datosJugador[2].setNombre(datos[7]);
			LvLJ3=datos[8];
			arrayCartas = datos[9].split(":");
			barajaJ3= newint[arrayCartas.length];
			for(int i=0;i<arrayCartas.length;i++){
```

```
                        barajaJ3[i]=Integer.parseInt(arrayCartas[i]);

                  }
                  datosJugador[2].setBaraja(barajaJ3);

                  if(numJugadores<4)return ;
                  //datos j4
                  Jugador4=datos[10];
                  datosJugador[3]=new DatosJugador();
                  datosJugador[3].setNombre(datos[10]);
                  LvLJ4=datos[11];
                  arrayCartas = datos[12].split(":");
                  barajaJ4= newint[arrayCartas.length];
                  for(int i=0;i<arrayCartas.length;i++){
                        barajaJ4[i]=Integer.parseInt(arrayCartas[i]);

                  }
                  datosJugador[3].setBaraja(barajaJ4);


                  System.out.println("tamano de barajas= "
+barajaJ1.length+" "+barajaJ2.length);

            }



            if(tipo.equals(Parser_TCP.SERV_PARTIDA_DADOS)){
                  dados=datos[0];
            }

            if(tipo.equals(Parser_TCP.SERV_PARTIDA_MANO_INI)){
                  jugador=datos[0];
                  carta=Integer.parseInt(datos[1]);
                  seleccion=Integer.parseInt(datos[2]);
            }
            if(tipo.equals(Parser_TCP.SERV_PARTIDA_JUGADA)){
                  jugador=datos[0];
                  carta=Integer.parseInt(datos[1]);
                  seleccion=Integer.parseInt(datos[2]);

            }if(tipo.equals(Parser_TCP.SERV_PARTIDA_FIN_RONDA)){
                  numJugadores=Integer.parseInt(datos[0]);
                  ganador=datos[1];
                  puntos=Integer.parseInt(datos[2]);

                  Jugador1=datos[3];
                  cartaJ1=Integer.parseInt(datos[4]);
                  selJ1=Integer.parseInt(datos[5]);

                  Jugador2=datos[6];
                  cartaJ2=Integer.parseInt(datos[7]);
                  selJ2=Integer.parseInt(datos[8]);

                  if(numJugadores>=3){
                        Jugador3=datos[9];
                        cartaJ3=Integer.parseInt(datos[10]);
                        selJ3=Integer.parseInt(datos[11]);
                  }
```

```
                    if(numJugadores>=4){
                          Jugador4=datos[12];
                          cartaJ4=Integer.parseInt(datos[13]);
                          selJ4=Integer.parseInt(datos[14]);
                    }

             }
             if(tipo.equals(Parser_TCP.SERV_PARTIDA_FIN)){
                    // demomento solo paradosjugadores
                    //numJugadores=Integer.parseInt(datos[0]);
                    ganador=datos[0];

                    Jugador1=datos[2];
                    puntosJ1=Integer.parseInt(datos[3]);
                    System.out.print("Jugador1 "+Jugador1);
                    System.out.print("Jpuntos j1"+puntosJ1);

                    Jugador2=datos[4];
                    System.out.print("Jugador2 "+Jugador2);
                    puntosJ2=Integer.parseInt(datos[5]);
                    System.out.print("Jpuntos j2 "+puntosJ2);


             }


       }
       public String getUsuario(){
             returnjugador;
       }
       publicint getCarta(){
             returncarta;
       }
       publicint getSeleccion(){
             returnseleccion;
       }
public String getGanador(){
       returnganador;
}

       publicvoid setTipo(String tipo){
             this.tipo=tipo;
       }
       public String getTipoMensaje(){
             returntipo;
       }
       public String[] getDatos(){
             returndatos;
       }
       publicint getPuntos(){
             returnpuntos;
       }

       publicint getNumJugadores(){
             returnnumJugadores;
       }

       public String getJugador1(){
```

```
		return Jugador1;
	}
	public String getJugador2(){
		return Jugador2;
	}
	public String getJugador3(){
		return Jugador3;
	}
	public String getJugador4(){
		return Jugador4;
	}
	publicint[] getBarajaJ1(){
		return barajaJ1;
	}
	publicint[] getBarajaJ2(){
		return barajaJ2;
	}
	publicint[] getBarajaJ3(){
		return barajaJ3;
	}
	publicint[] getBarajaJ4(){
		return barajaJ4;
	}
	public String getTirada(){
		returnthis.dados;
	}

}
```

Archivo DirectorOnline.java

```
ackage xnetcom.pro.cartas.red;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.SocketException;
import java.util.ArrayList;
import java.util.concurrent.Semaphore;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.auxiliares.DatosJugador;
import xnetcom.pro.cartas.auxiliares.datosUsuario;
import xnetcom.pro.cartas.juego.DirectorJuego;
import xnetcom.pro.cartas.jugadores.Jugador;
import xnetcom.pro.cartas.jugadores.JugadorHumano;
import xnetcom.pro.cartas.jugadores.JugadorRemoto;

publicclass DirectorOnline extends Thread{



	private Jugador[] jugadores;

	private datosUsuario datos;
	private Game juego;
	private Semaphore semaforo;
	privatebooleanerror=false;


	// cosasparalaconexion

protected DataOutputStream dos;
protected DataInputStream dis;
private String mensaje = "";
	private Socket cliente;
	//private String servidor = "192.168.2.101";
	private String servidor = "138.100.49.9";
	privateintPuerto=1234;
	private Datos_TCP datos_tcp;
	private Datos_TCP datos_tcp_ini;
private Parser_TCP parser;
private Sincro sincro;
private DirectorJuego DJ;
	private DatosJugador[] datosJugadores;
	private ArrayList<Datos_TCP>mensajes;
	public DirectorOnline(Game juego, DirectorJuego DJ){
		semaforo =new Semaphore(0);
		parser = new Parser_TCP();
		this.juego=juego;
		this.DJ=DJ;
```

```
			sincro =new Sincro();
			datos=new datosUsuario(juego);
			datos.leer_config();
		}

		publicint NuevaPartida(){
			System.out.println("NuevaPartida()");
			this.start();
			System.out.println("NuevaPartida()1");
			try {
				semaforo.acquire();
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			System.out.println("NuevaPartida()1.5");



			if (error){
				System.out.println("eerrorrrrr");
				juego.finalizar();
				return -1;

			}

			sincro.EsperaIniciado();
			System.out.println("NuevaPartida()2");
			mensajes=new ArrayList<Datos_TCP>();
			System.out.println("NuevaPartida()3");
			enviarCLI_JUGAR(datos.getUsuario(),datos.getPassword());
			System.out.println("NuevaPartida()4");
			sincro.EsperaEncolado();
			System.out.println("NuevaPartida()5");
			if
(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_ENCOLADO)){
				return 1;
			}
			System.out.println("retornamos un -1");
			return -1;
		}


		public Sincro getSincro(){
			returnsincro;
		}
		public DatosJugador[] getDatosJugadores(){
			returndatosJugadores;
		}
		public Datos_TCP getDatosTCP_ini(){
			returndatos_tcp_ini;
		}
		public Datos_TCP getDatosTCP(){
			returndatos_tcp;
		}

		publicvoid ejecutarCliente() {
			// conectarsealservidor, obtenerflujos, procesarlaconexión
			try {
```

```
                    conectarAServidor();      // Paso 1: crearun socket
pararealizarlaconexión
                    obtenerFlujos();          // Paso 2:
obtenerlosflujosdeentrada y salida
                    procesarConexion();       // Paso 3: procesarlaconexión
              }
              // el servidorcerrólaconexión
              catch (EOFException excepcionEOF) {
                    System.err.println("El cliente termino la conexión");
              }

              // procesarlosproblemasquepuedenocurriralcomunicarsecon el
              // servidor
              catch (IOException excepcionES) {
                    excepcionES.printStackTrace();
              } finally {
                    cerrarConexion(); // Paso 4: cerrarlaconexión
                    error=true;
                    semaforo.release();
              }
       } // fin delmétodo ejecutarCliente


       privatevoid conectarAServidor() throws IOException {
              // crear Socket pararealizarlaconexióncon el servidor
              cliente = new Socket(InetAddress.getByName(servidor), Puerto);
       }

       privatevoid obtenerFlujos() throws IOException {
              // establecerflujodesalidaparalosobjetos
              dos = new DataOutputStream(cliente.getOutputStream());
              dos.flush();
              // establecerflujodeentradaparalosobjetos
              dis = new DataInputStream(cliente.getInputStream());
              sincro.setIniciado();
       }

       public Jugador buscaJugadorPorUsuario(String usuario){
              System.out.println("soy buscador por usuario queren que buske
"+usuario);
              for(int i=0;i<jugadores.length;i++){
                    if (jugadores[i].getNombre().equals(usuario)){
                           System.out.println("soy buscador por usuario
jugadores[i] "+jugadores[i].getNombre());
                           returnjugadores[i];
                    }
              }
              System.out.println("soy buscador por usuario retorno null ");
              returnnull;
       }
       privatevoid procesarConexion() throws IOException {
              boolean desconexion =false;// puestodepegotemirarluego
              do {
                    try {
                           mensaje="noooorrrr";
                           mensaje = dis.readUTF();
                           System.out.println("mensaje reciiii"+mensaje);
                           datos_tcp = parser.nuevoMensaje(mensaje);
                           if (datos_tcp !=null){
```

```
	if(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_INI)){

					datos_tcp_ini=datos_tcp;
					SERV_PARTIDA_INI();
				}

	elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_DADOS
)){
					sincro.LlegadoDados();
				}

	elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_MANO_
INI)){
					JugadorRemoto JR=(JugadorRemoto)
buscaJugadorPorUsuario(datos_tcp.getUsuario());

	JR.getAccion().Accion(datos_tcp.getCarta(), datos_tcp.getSeleccion());
				}

	elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_JUGAD
A)){
					JugadorRemoto JR=(JugadorRemoto)
buscaJugadorPorUsuario(datos_tcp.getUsuario());

	JR.getAccion().Accion(datos_tcp.getCarta(), datos_tcp.getSeleccion());

	}elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_ENCOLADO)){
					sincro.Encolado();

	}elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_FIN_
RONDA)){
					sincro.Llegado_Fin_Ronda();

				}

	elseif(datos_tcp.getTipoMensaje().equals(Parser_TCP.SERV_PARTIDA_FIN))
{
					System.out.print("SERV_PARTIDA_FIN");

				}
				else {
					mensajes.add(datos_tcp);

					sincro.LlegadoMensaje();
				}



			}
		}

		catch (ParserConfigurationException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (SAXException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
```

```
                }
                catch (SocketException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        //desconexion=true;
                        juego.tostar("desconectado del servidor");

                        System.out.println("SocketException e");
                        return;
                }
                System.out.println("llego al while !desconexxion");
        }while (!desconexion);    // leer mensaje y mostrarloenpantalla

} // fin delmétodo procesarConexion


// cerrarflujos y socket
publicvoid cerrarConexion() {
        try {
                dis.close();
                dos.close();
                cliente.close();
        } catch (IOException excepcionES) {
                excepcionES.printStackTrace();
                System.out.println("IOException excepcionES");
                juego.tostar("error de conexion");
        }
        catch (Exception excepcionES) {
                excepcionES.printStackTrace();
                juego.tostar("error de conexion");
                System.out.println("Exception excepcionESsaaa");
                juego.tostar("error al conectarse al servidor");
                //juego.finish();

        }
        System.out.println("cerrados todos los sockets");
}

privatevoid enviarDatos(String mensaje) {
        // enviarobjetoalservidor
        try {
                System.out.println("envioooooo"+mensaje);
                dos.writeUTF(mensaje);
                dos.flush();
        }
        // procesarlosproblemasquepuedenocurriralenviar el objeto
        catch (IOException excepcionES) {
        }
}
publicvoid SERV_PARTIDA_INI(){


        datosJugadores=datos_tcp_ini.getDatosJugadores();
        DatosJugador datosJugadorTemp;

        //si el jugador que es el usuario no es el jugador 0 se ordena
        if(!datosJugadores[0].getNombre().equals(datos.getUsuario())){
                datosJugadorTemp=datosJugadores[0];
                for(int i=1;i<datosJugadores.length;i++){
```

```
      if(datosJugadores[i].getNombre().equals(datos.getUsuario())){
                                    datosJugadores[0]=datosJugadores[i];
                                    datosJugadores[i]=datosJugadorTemp;
                                    i=datosJugadores.length;
                              }
                        }
                  }
                  jugadores = new Jugador[datos_tcp_ini.getNumJugadores()];

                  jugadores[0] = new JugadorHumano(juego, 1, 1,
datos.getUsuario());


      System.out.println("numjugadores"+datos_tcp_ini.getNumJugadores());
                  for (int i=1;i<datos_tcp_ini.getNumJugadores();i++){
                        jugadores[i] = new JugadorRemoto(juego, 3, i + 1,
datosJugadores[i].getNombre());
                  }

                  DJ.setJugadores(jugadores);
                  sincro.InicadoPartida();
      }

      //mensajes de salida


      publicvoid enviarCLI_JUGAR(String... parametros){

                  StringBuffer output = new StringBuffer();
                  output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
                  output.append("<mensaje protocol_ver=\"1.0\">");
                  output.append("<mensaje_ID>CLI_JUGAR</mensaje_ID>");
                  output.append("<datos>");
                  output.append("<usuario>" + parametros[0]+ "</usuario>");
                  output.append("<password>" + parametros[1]+ "</password>");
                  output.append("<baraja>" + "1.0"+ "</baraja>");
                  output.append("</datos>");
                  output.append("</mensaje>");
                  enviarDatos(output.toString());

      }

      //parametros nombre del jugador
                  publicvoid enviarCLI_PARTIDA_INI_ACK(){
                        StringBuffer output = new StringBuffer();
                        output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
                        output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_PARTIDA_INI_ACK</mensaje_ID>");
                        output.append("<datos>");

      output.append("<usuario>"+datos.getUsuario()+"</usuario>");
                        output.append("</datos>");
                        output.append("</mensaje>");
                        enviarDatos(output.toString());
                  }
```

```
            // parametros usuario, carta, seleccion
            publicvoid enviarCLI_PARTIDA_MANO_INI(String... parametros) {
                  StringBuffer output = new StringBuffer();
                  output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
                  output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_PARTIDA_MANO_INI</mensaje_ID>");
                  output.append("<datos>");
                  output.append("<usuario>"+parametros[0]+ "</usuario>");
                  output.append("<carta>"+parametros[1]+ "</carta>");
                  output.append("<seleccion>"+parametros[2]+
"</seleccion>");
                  output.append("</datos>");
                  output.append("</mensaje>");
                  enviarDatos(output.toString());
            }


            // parametros usuario,carta, seleccion
            publicvoid enviarCLI_PARTIDA_JUGADA(String... parametros) {
                  StringBuffer output = new StringBuffer();
                  output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
                  output.append("<mensaje protocol_ver=\"1.0\">");

      output.append("<mensaje_ID>CLI_PARTIDA_JUGADA</mensaje_ID>");
                  output.append("<datos>");
                  output.append("    <usuario>"+parametros[0]+
"</usuario>");
                  output.append("    <carta>"+parametros[1]+ "</carta>");
                  output.append("    <seleccion>"+parametros[2]+
"</seleccion>");
                  output.append("</datos>");
                  output.append("</mensaje>");
                  enviarDatos(output.toString());
            }

            // usuario contraseña
            publicvoid enviarCLI_DESCONECTAR(String... parametros){
                  StringBuffer output = new StringBuffer();
                  output.append("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>");
                  output.append("<mensaje protocol_ver=\"1.0\">");
                  output.append("<mensaje_ID>CLI_DESCONECTAR</mensaje_ID>");
                  output.append("<usuario>" + datos.getUsuario()+
"</usuario>");
                  output.append("<password>" + datos.getPassword()+
"</password>");
                  output.append("</mensaje>");
                  enviarDatos(output.toString());
            }



      publicvoid run(){
            ejecutarCliente();
```

```
	}

}
```

Archivo HiloTCP.java

```
package xnetcom.pro.cartas.red;

import org.anddev.andengine.ui.activity.BaseGameActivity;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.activities.MainMenu;

publicclass HiloTCP extends Thread {

	private MainMenu menu=null;
	private Game juego=null;
	private clienteTCP cli;

	public HiloTCP(BaseGameActivity contexto) {
		super();
		if (contexto instanceof MainMenu){
			menu =(MainMenu) contexto;
			cli = new clienteTCP(menu,this);
		}
		if (contexto instanceof Game){
			juego =(Game) contexto;
			cli = new clienteTCP(juego,this);
		}



		// TODO Auto-generated constructor stub
	}

	publicvoid run()
	{

		cli.ejecutarCliente();
		System.out.println("soy el hilo salgo del run");
	}

	publicvoid desconectar(){
		System.out.println("\nsoy el hilo ctp intento desconectar");
		cli.desconexion();
		//this.interrupt();
		//this.notify();
	}

	public clienteTCP getclienteTCP(){
		returncli;
	}
}
```

Archivo ParserTCP.java

```
package xnetcom.pro.cartas.red;


import java.io.IOException;
import java.io.StringReader;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;


publicclass Parser_TCP {

/*
 * Mensajesmandadosdesde el servidor
*
 */

	publicstatic String SERV_ENCOLADO="SERV_ENCOLADO";
	publicstatic String SERV_DESCONECTAR_ACK ="SERV_DESCONECTAR_ACK";
	//public static String SERV_LOG_OK="SERV_LOG_OK";
	publicstatic String SERV_REGISTRAR_OK="SERV_REGISTRAR_OK";
	publicstatic String SERV_LOG_ERROR="SERV_LOG_ERROR";
	publicstatic String SERV_MOSTRAR_PERFIL="SERV_MOSTRAR_PERFIL";
	publicstatic String SERV_STAT="SERV_STAT";
	publicstatic String SERV_PARTIDA_INI="SERV_PARTIDA_INI";
	publicstatic String SERV_PARTIDA_DADOS="SERV_PARTIDA_DADOS";
	publicstatic String SERV_PARTIDA_MANO_INI="SERV_PARTIDA_MANO_INI";
	publicstatic String SERV_PARTIDA_JUGADA="SERV_PARTIDA_JUGADA";
	publicstatic String SERV_PARTIDA_FIN_RONDA="SERV_PARTIDA_FIN_RONDA";
	publicstatic String SERV_PARTIDA_FIN="SERV_PARTIDA_FIN";
	publicstatic String SERV_CHAT="SERV_CHAT";
	publicstatic String SERV_MOSTAR_TEXTO="SERV_MOSTAR_TEXTO";
	publicstatic String SERV_ERROR="SERV_ERROR";
	publicstatic String SERV_MOD_OK="SERV_MOD_OK";


	private NodeList parametros=null;

	private String datos[];
	private String tipo_mensaje;

	public Datos_TCP nuevoMensaje(String mensaje) throws
ParserConfigurationException, SAXException, IOException  {

		datos= new String[20];
		this.tipo_mensaje = "no valido";
		System.out.println("nuevo mensaje :"+mensaje);
		try{
		System.out.println("nuevo mensaje partido
:"+mensaje.split("<baraja>")[1]);
```

```
            }catch(Exception e){}


            DocumentBuilder builder;

            if (mensaje.startsWith("<?xml version=\"1.0\" encoding=\"UTF-8\"
standalone=\"no\"?>")) {
                  System.out.println("llegado mensaje acorde con la
cabezera");
            //if (mensaje.startsWith("<?xml version=\"1.0\" encoding=\"iso-
8859-1\"?>")) {

                  DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
                  builder = factory.newDocumentBuilder();
                  Document dom = builder.parse(new InputSource(new
StringReader(mensaje)));

                  Element root = dom.getDocumentElement();
                  NodeList mensajes = root.getElementsByTagName("mensaje");
                  Node nodo = mensajes.item(0);

                  Document documento = builder.parse(new InputSource(new
StringReader(mensaje)));

      NodeList listaNodos = documento.getElementsByTagName("mensaje_ID");
      for (int i = 0; i < listaNodos.getLength(); i++) {
                 System.out.println("metodo comprobar tipo mensaje " +
listaNodos.item(i).getTextContent());

System.out.println("listanodos"+listaNodos.item(i).getTextContent());
      tipo_mensaje=listaNodos.item(i).getTextContent();

              }

                  //tipo_mensaje =
nodo.getFirstChild().getNextSibling().getTextContent().trim();

                  System.out.println("tipomensaje:"+tipo_mensaje);
                  parametros=null;
                  try{
                        NodeList datos =
root.getElementsByTagName("datos");
                        Element dat = (Element) datos.item(0);
                        parametros = dat.getChildNodes();
                  }catch(Exception e){}


                  if (tipo_mensaje.equals(SERV_ENCOLADO)) {
                        System.out.println("encolado");
                  }
                  if (tipo_mensaje.equals(SERV_DESCONECTAR_ACK)) {
                        System.out.println("ERV_DESCONECTAR_ACK");
                  }
                  /*
                  if (tipo_mensaje.equals(SERV_LOG_OK)) {
                        System.out.println("SERV_LOG_OK");
                  }
                  */
```

```
                    if (tipo_mensaje.equals(SERV_LOG_ERROR)) {
                          System.out.println("SERV_LOG_ERROR");
                    }
                    if (tipo_mensaje.equals(SERV_MOSTRAR_PERFIL)) {

                          System.out.println("es un mostrar perfil");
                          SERV_MOSTRAR_PERFIL();
                    }

                    if (tipo_mensaje.equals(SERV_ERROR)) {
                          SERV_ERROR();
                    }
                    if (tipo_mensaje.equals(SERV_MOD_OK)) {

                          SERV_MOD_OK();
                    }
                    if (tipo_mensaje.equals(SERV_STAT)) {

                          inti=0;

                          for (int j = 0; j <parametros.getLength(); j++) {



                          }
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_INI)) {
                          this.SERV_PARTIDA_INI();
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_DADOS)) {

                          SERV_PARTIDA_DADOS();
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_MANO_INI)) {

                          SERV_PARTIDA_MANO_INI();
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_JUGADA)) {

                          SERV_PARTIDA_JUGADA();
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_FIN_RONDA)) {

                          SERV_PARTIDA_FIN_RONDA();
                    }
                    if (tipo_mensaje.equals(SERV_PARTIDA_FIN)) {

                          SERV_PARTIDA_FIN();
                    }
                    if (tipo_mensaje.equals(SERV_CHAT)) {
                          for (int j = 0; j <parametros.getLength(); j++) {
                                if
(parametros.item(j).getNodeName().trim().equals("usuario")) {

      System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
                                }
                                if
(parametros.item(j).getNodeName().trim().equals("chat")) {
```

```
      System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
                              }
                        }

                  }
                  if (tipo_mensaje.equals(SERV_MOSTAR_TEXTO)) {
                        for (int j = 0; j <parametros.getLength(); j++) {
                              if
(parametros.item(j).getNodeName().trim().equals("texto")) {

      System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
                              }
                        }

                  }
            } else {
                  System.out.println("llegado un mensaje no acorde con la
cabecera");
                  returnnull;
            }

            Datos_TCP retorno = new Datos_TCP(this.tipo_mensaje);
            retorno.setDatos(datos);
            System.out.println("datos de retorno del
parser"+retorno.getDatos()[0]);
            return retorno;

      }

      publicvoid SERV_PARTIDA_FIN(){



            for (int j = 0; j <parametros.getLength(); j++) {

      if(parametros.item(j).getNodeName().trim().equals("ganador")){
                        datos[0]=parametros.item(j).getTextContent();

      System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());

                  }


      if(parametros.item(j).getNodeName().trim().equals("jugador1")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[2]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(puntos)
"+jugador.getElementsByTagName("puntos").item(0).getTextContent());
```

```
      datos[3]=jugador.getElementsByTagName("puntos").item(0).getTextContent
();

                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador2")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[4]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(puntos)
"+jugador.getElementsByTagName("puntos").item(0).getTextContent());

      datos[5]=jugador.getElementsByTagName("puntos").item(0).getTextContent
();
                        datos[1]="2";

                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador3")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[6]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(puntos)
"+jugador.getElementsByTagName("puntos").item(0).getTextContent());

      datos[7]=jugador.getElementsByTagName("puntos").item(0).getTextContent
();
                        datos[1]="3";
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador4")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[8]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(puntos)
"+jugador.getElementsByTagName("puntos").item(0).getTextContent());

      datos[9]=jugador.getElementsByTagName("puntos").item(0).getTextContent
();
                        datos[1]="4";

                  }
```

```
                    }

      }

      publicvoid SERV_PARTIDA_FIN_RONDA(){

            //int i=0;
            for (int j = 0; j <parametros.getLength(); j++) {

      if(parametros.item(j).getNodeName().trim().equals("num_jugadores")){
                        datos[0]=parametros.item(j).getTextContent();

      System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());

                    }

      if(parametros.item(j).getNodeName().trim().equals("ganador")){
                        datos[1]=parametros.item(j).getTextContent();

      System.out.println("parametros.item(j).getTextContent()ganador;"+param
etros.item(j).getTextContent());

                    }

      if(parametros.item(j).getNodeName().trim().equals("puntos")){
                        datos[2]=parametros.item(j).getTextContent();

      System.out.println("parametros.item(j).getTextContent()puntos;"+parame
tros.item(j).getTextContent());

                    }


      if(parametros.item(j).getNodeName().trim().equals("jugador1")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[3]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(carta)
"+jugador.getElementsByTagName("carta").item(0).getTextContent());

      datos[4]=jugador.getElementsByTagName("carta").item(0).getTextContent(
);

      System.out.println("jugador.getElementsByTagName(seleccion)
"+jugador.getElementsByTagName("seleccion").item(0).getTextContent());

      datos[5]=jugador.getElementsByTagName("seleccion").item(0).getTextCont
ent();
                    }

      if(parametros.item(j).getNodeName().trim().equals("jugador2")){
                        Element jugador =(Element)parametros.item(j);
```

```
      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[6]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(carta)
"+jugador.getElementsByTagName("carta").item(0).getTextContent());

      datos[7]=jugador.getElementsByTagName("carta").item(0).getTextContent(
);

      System.out.println("jugador.getElementsByTagName(seleccion)
"+jugador.getElementsByTagName("seleccion").item(0).getTextContent());

      datos[8]=jugador.getElementsByTagName("seleccion").item(0).getTextCont
ent();
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador3")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[9]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();

      System.out.println("jugador.getElementsByTagName(carta)
"+jugador.getElementsByTagName("carta").item(0).getTextContent());

      datos[10]=jugador.getElementsByTagName("carta").item(0).getTextContent
();

      System.out.println("jugador.getElementsByTagName(seleccion)
"+jugador.getElementsByTagName("seleccion").item(0).getTextContent());

      datos[11]=jugador.getElementsByTagName("seleccion").item(0).getTextCon
tent();
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador4")){
                        Element jugador =(Element)parametros.item(j);

      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[12]=jugador.getElementsByTagName("usuario").item(0).getTextConte
nt();

      System.out.println("jugador.getElementsByTagName(carta)
"+jugador.getElementsByTagName("carta").item(0).getTextContent());

      datos[13]=jugador.getElementsByTagName("carta").item(0).getTextContent
();

      System.out.println("jugador.getElementsByTagName(seleccion)
"+jugador.getElementsByTagName("seleccion").item(0).getTextContent());
```

```
      datos[14]=jugador.getElementsByTagName("seleccion").item(0).getTextCon
tent();
                  }


            }

      }

      publicvoid SERV_PARTIDA_DADOS(){
            for (int j = 0; j <parametros.getLength(); j++) {
                  if
(parametros.item(j).getNodeName().trim().equals("turno")) {
                        datos[0]=parametros.item(j).getTextContent();

      System.out.println("parametros.item(j).getTextContent();"+
parametros.item(j).getTextContent());
                  }
            }
      }

      publicvoid SERV_ERROR(){

            for (int j = 0; j <parametros.getLength(); j++) {

      if(parametros.item(j).getNodeName().trim().equals("texto")){
                        System.out.println("soy parser tcp  miro texto");
                        datos[0]=parametros.item(j).getTextContent();
                  }
            }

      }


      publicvoid SERV_PARTIDA_MANO_INI(){

            for (int j = 0; j <parametros.getLength(); j++) {

      if(parametros.item(j).getNodeName().trim().equals("turno")){


      System.out.println("parametros.item(j).getTextContent()(turno);"+param
etros.item(j).getTextContent());
                        datos[0]=parametros.item(j).getTextContent();
                  }

      if(parametros.item(j).getNodeName().trim().equals("carta")){


      System.out.println("parametros.item(j).getTextContent()(carta);"+param
etros.item(j).getTextContent());
                        datos[1]=parametros.item(j).getTextContent();
                  }

      if(parametros.item(j).getNodeName().trim().equals("seleccion")){
```

```
	System.out.println("parametros.item(j).getTextContent();(seleccion)"+p
arametros.item(j).getTextContent());
				datos[2]=parametros.item(j).getTextContent();
			}
		}
	}

	publicvoid SERV_PARTIDA_JUGADA(){
		for (int j = 0; j <parametros.getLength(); j++) {

	if(parametros.item(j).getNodeName().trim().equals("turno")){


	System.out.println("parametros.item(j).getTextContent()(turno);"+param
etros.item(j).getTextContent());
				datos[0]=parametros.item(j).getTextContent();
			}

	if(parametros.item(j).getNodeName().trim().equals("carta")){


	System.out.println("parametros.item(j).getTextContent()(carta);"+param
etros.item(j).getTextContent());
				datos[1]=parametros.item(j).getTextContent();
			}

	if(parametros.item(j).getNodeName().trim().equals("seleccion")){


	System.out.println("parametros.item(j).getTextContent();(seleccion)"+p
arametros.item(j).getTextContent());
				datos[2]=parametros.item(j).getTextContent();
			}
		}
	}

	publicvoid SERV_PARTIDA_INI(){


		int i=0;

		for (int j = 0; j <parametros.getLength(); j++) {

	if(parametros.item(j).getNodeName().trim().equals("num_jugadores")){
				datos[i]=parametros.item(j).getTextContent();

	System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
				i++;//0
			}


	if(parametros.item(j).getNodeName().trim().equals("jugador1")){

				Element jugador =(Element)parametros.item(j);
```

```
      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();i++;     //1


      System.out.println("jugador.getElementsByTagName(nivel)
"+jugador.getElementsByTagName("nivel").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("nivel").item(0).getTextContent(
);i++; //2


      System.out.println("jugador.getElementsByTagName(baraja)
"+jugador.getElementsByTagName("baraja").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("baraja").item(0).getTextContent
();i++;      //3
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador2")){
                        Element jugador =(Element)parametros.item(j);


      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();i++;     //4


      System.out.println("jugador.getElementsByTagName(nivel)
"+jugador.getElementsByTagName("nivel").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("nivel").item(0).getTextContent(
);i++; //5


      System.out.println("jugador.getElementsByTagName(baraja)
"+jugador.getElementsByTagName("baraja").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("baraja").item(0).getTextContent
();i++;      //6
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador3")){
                        Element jugador =(Element)parametros.item(j);


      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();i++;     //7
```

```
      System.out.println("jugador.getElementsByTagName(nivel)
"+jugador.getElementsByTagName("nivel").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("nivel").item(0).getTextContent(
);i++; //8


      System.out.println("jugador.getElementsByTagName(baraja)
"+jugador.getElementsByTagName("baraja").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("baraja").item(0).getTextContent
();i++;      //9
                  }

      if(parametros.item(j).getNodeName().trim().equals("jugador4")){
                        Element jugador =(Element)parametros.item(j);


      System.out.println("jugador.getElementsByTagName(usuario)
"+jugador.getElementsByTagName("usuario").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("usuario").item(0).getTextConten
t();i++;     //10


      System.out.println("jugador.getElementsByTagName(nivel)
"+jugador.getElementsByTagName("nivel").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("nivel").item(0).getTextContent(
);i++; //11


      System.out.println("jugador.getElementsByTagName(baraja)
"+jugador.getElementsByTagName("baraja").item(0).getTextContent());

      datos[i]=jugador.getElementsByTagName("baraja").item(0).getTextContent
();i++;      //12
                  }

            }
      }

      publicvoid SERV_MOD_OK(){
      }

      publicvoid SERV_MOSTRAR_PERFIL(){
            //this.tipo_mensaje=SERV_MOSTAR_PERFIL;
            System.out.println("soy parser tcp recivido mensaje de perfil");
            int i=0;

            for (int j = 0; j <parametros.getLength(); j++) {


      if(parametros.item(j).getNodeName().trim().equals("usuario")){
                        datos[i]=parametros.item(j).getTextContent();
                        i++;
                  }
```

```
	if(parametros.item(j).getNodeName().trim().equals("nombre")){
				datos[i]=parametros.item(j).getTextContent();

				i++;
			}

	if(parametros.item(j).getNodeName().trim().equals("apellidos")){
				datos[i]=parametros.item(j).getTextContent();
				i++;
			}

	if(parametros.item(j).getNodeName().trim().equals("pais")){
				datos[i]=parametros.item(j).getTextContent();
				i++;
			}

	if(parametros.item(j).getNodeName().trim().equals("avatar")){
				datos[i]=parametros.item(j).getTextContent();
				i++;
			}

	if(parametros.item(j).getNodeName().trim().equals("nivel")){
				datos[i]=parametros.item(j).getTextContent();
				i++;

	System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
			}

	if(parametros.item(j).getNodeName().trim().equals("jugadas")){
				datos[i]=parametros.item(j).getTextContent();
				i++;

	System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
			}

	if(parametros.item(j).getNodeName().trim().equals("ganadas")){
				datos[i]=parametros.item(j).getTextContent();
				i++;

	System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
			}

	if(parametros.item(j).getNodeName().trim().equals("puntos")){
				datos[i]=parametros.item(j).getTextContent();
				i++;

	System.out.println("parametros.item(j).getTextContent();"+parametros.i
tem(j).getTextContent());
			}

		}

	}

}
```

Archivo Sincro.java

```
package xnetcom.pro.cartas.red;

publicclass Sincro {
	privatebooleansinc = false;
	privatebooleanencolado=false;
	privatebooleanMensaje=false;
	privatebooleandados=false;
	privatebooleanpartida=false;
	privatebooleanfinRonda=false;
	privateintnumMensajes=0;

	publicsynchronizedvoid EsperaIniciado() {
		try {
			while (!sinc) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	publicsynchronizedvoid setIniciado() {
		sinc=true;
		this.notify();
	}

	publicsynchronizedvoid Encolado() {
		encolado=true;
		this.notify();
	}

	publicsynchronizedvoid EsperaEncolado() {
		try {
			while (!encolado) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	publicsynchronizedvoid EsperaMensaje() {
		try {
			while (numMensajes==0) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		numMensajes--;
	}
	publicsynchronizedvoid LlegadoMensaje() {
		numMensajes++;
		this.notify();
```

```
	}


	publicsynchronizedvoid EsperaPartida() {
		try {
			while (!partida) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	publicsynchronizedvoid IniciadoPartida() {
		partida=true;
		this.notify();
	}


	publicsynchronizedvoid Espera_Fin_Ronda() {
		try {
			while (!finRonda) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		finRonda=false;
	}

	publicsynchronizedvoid Llegado_Fin_Ronda() {
		finRonda=true;
		this.notify();
	}
	publicsynchronizedvoid EsperaDados() {
		try {
			while (!dados) {
				this.wait();
			}
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		dados=false;
	}
	publicsynchronizedvoid LlegadoDados() {
		dados=true;
		this.notify();
	}

	publicsynchronizedvoid BoqueoMutex(){

	}


}
```

Archivo BlogSprite.java

```
package xnetcom.pro.cartas.sprites;


import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.text.ChangeableText;
import org.anddev.andengine.input.touch.TouchEvent;
import org.anddev.andengine.opengl.font.Font;
import org.anddev.andengine.opengl.texture.region.TextureRegion;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Carta;
import xnetcom.pro.cartas.jugadores.Jugador;

publicclass BlogSprite extends Sprite{

	private ChangeableText DatosJ1_1;
	private ChangeableText DatosJ1_2;
	private ChangeableText DatosJ1_3;
	private ChangeableText DatosJ1_4;
	private ChangeableText DatosJ2_1;
	private ChangeableText DatosJ2_2;
	private ChangeableText DatosJ2_3;
	private ChangeableText DatosJ2_4;
	private ChangeableText DatosJ3_1;
	private ChangeableText DatosJ3_2;
	private ChangeableText DatosJ3_3;
	private ChangeableText DatosJ3_4;
	private ChangeableText DatosJ4_1;
	private ChangeableText DatosJ4_2;
	private ChangeableText DatosJ4_3;
	private ChangeableText DatosJ4_4;
	private ChangeableText Datos_1;
	private ChangeableText Datos_2;
	private ChangeableText Datos_3;
	private ChangeableText Datos_4;
	private Game juego;

	public BlogSprite(Game juego,float pX, float pY, TextureRegion
pTextureRegion) {
		super(pX, pY, pTextureRegion);
		this.juego=juego;
		this.setZIndex(5000);
		Datos_1= new ChangeableText(60, 50+30, juego.getFont(), " ",
200);
		Datos_2= new ChangeableText(65+140, 50+30, juego.getFont(), " ",
200);
		Datos_3= new ChangeableText(65+390, 50+30, juego.getFont(), " ",
200);
		Datos_4= new ChangeableText(65+515, 50+30, juego.getFont(), " ",
200);
		Datos_1.setText("Jugador");
		Datos_2.setText("Pais");
		Datos_3.setText("Dato");
		Datos_4.setText("Valor");
		// TODO Auto-generated constructor stub
	}
```

```
	public Font colorGanador(int numero, int ganador){
		if (numero==ganador) returnjuego.getFontBlue();
		returnjuego.getFont();
	}


	publicvoid setBlogFinal(Jugador[] jugadores){
		int lon =jugadores.length;
		Datos_2.setText("Puntuacion Final");
		if
(lon>=1)DatosJ1_2.setText(Integer.toString(jugadores[0].puntos()));
		if
(lon>=2)DatosJ2_2.setText(Integer.toString(jugadores[1].puntos()));
		if
(lon>=3)DatosJ3_2.setText(Integer.toString(jugadores[2].puntos()));
		if
(lon>=4)DatosJ4_2.setText(Integer.toString(jugadores[3].puntos()));

	}
	public String addEspacios(String nombre, int longitud){
		String nom=nombre;
		int lon=nombre.length();
		int espacios=longitud-lon;
		System.out.println("addespacios :->" + nombre + "<-" +
espacios);
		for (int i = 0; i < espacios; i++) {
			nom+=" ";
		}
		System.out.println("despues: ->"+nom+"<-");
		return nom;
	}
	publicvoid ponerDatos(Carta j1, Carta j2, Carta j3, Carta j4, int
ganador){
		int sel=juego.getDirectorJuego().getCartaJuego().getSeleccion();
		String dato="nulo";
		switch (sel){

		case 1:dato="IDH      "; break;
		case 2:dato="Salud    "; break;
		case 3:dato="Educacion"; break;
		case 4:dato="Ingresos "; break;


		}
		this.detachChildren();
		//this.Datos= new ChangeableText(50, 70, juego.getFont(), " ",
200);
		this.DatosJ1_1= new ChangeableText(60, 50+30+50, colorGanador(1,
ganador), " ", 200);
		this.DatosJ2_1= new ChangeableText(60, 100+30+50,
colorGanador(2, ganador), " ", 200);
		this.DatosJ3_1= new ChangeableText(60, 150+30+50,
colorGanador(3, ganador), " ", 200);
		this.DatosJ4_1= new ChangeableText(60, 200+30+50,
colorGanador(4, ganador), " ", 200);

		this.DatosJ1_2= new ChangeableText(65+140, 50+30+50,
colorGanador(1, ganador), " ", 200);
```

```
            this.DatosJ2_2= new ChangeableText(65+140, 100+30+50,
colorGanador(2, ganador), " ", 200);
            this.DatosJ3_2= new ChangeableText(65+140, 150+30+50,
colorGanador(3, ganador), " ", 200);
            this.DatosJ4_2= new ChangeableText(65+140, 200+30+50,
colorGanador(4, ganador), " ", 200);

            this.DatosJ1_3= new ChangeableText(65+390, 50+30+50,
colorGanador(1, ganador), " ", 200);
            this.DatosJ2_3= new ChangeableText(65+390, 100+30+50,
colorGanador(2, ganador), " ", 200);
            this.DatosJ3_3= new ChangeableText(65+390, 150+30+50,
colorGanador(3, ganador), " ", 200);
            this.DatosJ4_3= new ChangeableText(65+390, 200+30+50,
colorGanador(4, ganador), " ", 200);

            this.DatosJ1_4= new ChangeableText(65+520, 50+30+50,
colorGanador(1, ganador), " ", 200);
            this.DatosJ2_4= new ChangeableText(65+520, 100+30+50,
colorGanador(2, ganador), " ", 200);
            this.DatosJ3_4= new ChangeableText(65+520, 150+30+50,
colorGanador(3, ganador), " ", 200);
            this.DatosJ4_4= new ChangeableText(65+520, 200+30+50,
colorGanador(4, ganador), " ", 200);


            Jugador [] jugadores=juego.getDirectorJuego().getJugadores();

            //             Jugador 1: EspañaDato    Valor

            DatosJ1_1.setText(jugadores[0].getNombre());
            DatosJ2_1.setText(jugadores[1].getNombre());
            if(j3!=null)DatosJ3_1.setText(jugadores[2].getNombre());
            if(j4!=null)DatosJ4_1.setText(jugadores[3].getNombre());


            DatosJ1_2.setText(j1.getNombre());
            DatosJ2_2.setText(j2.getNombre());
            if(j3!=null)DatosJ3_2.setText(j3.getNombre());
            if(j4!=null)DatosJ4_2.setText(j4.getNombre());


            DatosJ1_3.setText(dato);
            DatosJ2_3.setText(dato);
            if(j3!=null)DatosJ3_3.setText(dato);
            if(j4!=null)DatosJ4_3.setText(dato);


            DatosJ1_4.setText(""+j1.getValorSeleccion(sel));
            DatosJ2_4.setText(""+j2.getValorSeleccion(sel));
            if(j3!=null)DatosJ3_4.setText(""+j3.getValorSeleccion(sel));
            if(j4!=null)DatosJ4_4.setText(""+j4.getValorSeleccion(sel));

            //this.attachChild(Datos);
            this.attachChild(DatosJ1_1);
            this.attachChild(DatosJ2_1);
            if(j3!=null)this.attachChild(DatosJ3_1);
            if(j4!=null)this.attachChild(DatosJ4_1);
```

```
            this.attachChild(DatosJ1_2);
            this.attachChild(DatosJ2_2);
            if(j3!=null)this.attachChild(DatosJ3_2);
            if(j4!=null)this.attachChild(DatosJ4_2);

            this.attachChild(DatosJ1_3);
            this.attachChild(DatosJ2_3);
            if(j3!=null)this.attachChild(DatosJ3_3);
            if(j4!=null)this.attachChild(DatosJ4_3);

            this.attachChild(DatosJ1_4);
            this.attachChild(DatosJ2_4);
            if(j3!=null)this.attachChild(DatosJ3_4);
            if(j4!=null)this.attachChild(DatosJ4_4);

            this.attachChild(Datos_1);
            this.attachChild(Datos_2);
            this.attachChild(Datos_3);
            this.attachChild(Datos_4);


            this.setVisible(true);
            //juego.getScene().attachChild(this);
            //juego.getScene().registerTouchArea(this);
      }


      publicboolean onAreaTouched(final TouchEvent pSceneTouchEvent,
      finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
            System.out.println("tocado blogggg");
            if(pSceneTouchEvent.getAction()==0){
                  this.setVisible(false);
                  juego.getDirectorJuego().getCD().countDown();
                  //juego.getScene().detachChild(this);
            }

            returnfalse;
      }

}
```

Archivo Dado.java

```
package xnetcom.pro.cartas.sprites;

import java.util.Random;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.audio.sound.Sound;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
importorg.anddev.andengine.entity.sprite.AnimatedSprite.IAnimationListener;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.opengl.util.GLHelper;

publicclass Dado extends AnimatedSprite {

	private Thread thread;
	privateinttirada = 0;
	private Random generator;
	private Sound sonido;

	public Dado(float pX, float pY, TiledTextureRegion
pTiledTextureRegion,	Sound sonido) {
		super(pX, pY, pTiledTextureRegion);
		generator = new Random();
		this.sonido = sonido;
		// TODO Auto-generated constructor stub
	}

	protectedvoid applyRotation(final GL10 pGL) {
		// * Disable culling so we can see the backside of this sprite.
		GLHelper.disableCulling(pGL);
		finalfloat rotation = this.mRotation;
		if (rotation != 0) {
			finalfloat rotationCenterX = this.mRotationCenterX;
			finalfloat rotationCenterY = this.mRotationCenterY;

			pGL.glTranslatef(rotationCenterX, rotationCenterY, 0);
			pGL.glRotatef(rotation, 1, 0, 0);
			pGL.glTranslatef(-rotationCenterX, -rotationCenterY, 0);
		}
	}

	publicint tirarDado(int numJugadores) {
		setVisible(true);
		sonido.play();
		sonido.setVolume(5f);
		System.out.println("volumen" + sonido.getVolume());
		int ran = tomaDecision(0, numJugadores-1);
		System.out.println("Aleatorio" + ran);
		for (int i = 80; i < 280; i = i + 40) {
			animate(i);
			try {
				Thread.currentThread().sleep(500);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
```

```
			stopAnimation();
			setCurrentTileIndex(ran);
			setTirada(getCurrentTileIndex() + 1);
			try {
				Thread.currentThread().sleep(2000);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			setVisible(false);
			return getTirada();
		}

		publicint tirarDadoConResultado(int resultado) {
			setVisible(true);
			sonido.play();
			sonido.setVolume(5f);
			System.out.println("volumen" + sonido.getVolume());


			for (int i = 80; i < 280; i = i + 40) {
				animate(i);
			try {
					Thread.currentThread().sleep(500);
				} catch (InterruptedException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
			stopAnimation();
			setCurrentTileIndex(resultado-1);
			setTirada(getCurrentTileIndex() + 1);
			try {
				Thread.currentThread().sleep(2000);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			setVisible(false);
			return getTirada();
		}

		publicint getTirada() {
			returntirada;
		}

		publicvoid setTirada(int tirada) {
			this.tirada = tirada;
		}

		privateint tomaDecision(int aStart, int aEnd) {

			long range = (long) aEnd - (long) aStart + 1;
			long fraction = (long) (range * generator.nextDouble());
			int aleatorio = (int) (fraction + aStart);
			return aleatorio;
		}

}
```

Archivo MiBaseSprite.java

```
package xnetcom.pro.cartas.sprites;


import org.anddev.andengine.entity.sprite.BaseSprite;
import org.anddev.andengine.opengl.texture.region.BaseTextureRegion;
import org.anddev.andengine.opengl.texture.region.TextureRegion;



publicabstractclass MiBaseSprite extends BaseSprite {
	// ===========================================================
	// Constructors
	// ===========================================================


	public MiBaseSprite(float pX, float pY, float pWidth, float pHeight,
			BaseTextureRegion pTextureRegion) {
		super(pX, pY, pWidth, pHeight, pTextureRegion);
		// TODO Auto-generated constructor stub
	}

	// ===========================================================
	// Constants
	// ===========================================================

	// ===========================================================
	// Fields
	// ===========================================================

	protected BaseTextureRegion mTextureRegion;
	protected BaseTextureRegion mTextureRegionDorso;
	protectedfloatmZ=0;
	protectedintrX=0;
	protectedintrY=1;
	protectedintrZ=0;




	// ===========================================================
	// Getter & Setter
	// ===========================================================

	public BaseTextureRegion getTextureRegion() {
		returnthis.mTextureRegion;
	}
	publicvoid setTextureRegion(TextureRegion pTextureRegion) {
		this.mTextureRegion=pTextureRegion;
	}

	publicvoid setFlippedHorizontal(finalboolean pFlippedHorizontal) {
		this.mTextureRegion.setFlippedHorizontal(pFlippedHorizontal);
	}

	publicvoid setFlippedVertical(finalboolean pFlippedVertical) {
		this.mTextureRegion.setFlippedVertical(pFlippedVertical);
	}
```

```
	// ============================================================
	// Methods for/from SuperClass/Interfaces
	// ============================================================


	// ============================================================
	// Methods
	// ============================================================



	publicvoid translationZ(float z) {
		this.mZ=z;
	}

	protectedfloat getEjeZ(){
		returnthis.mZ;
	}


	// ============================================================
	// Inner and Anonymous Classes
	// ============================================================
}
```

Archivo MiSprite,java

```
package xnetcom.pro.cartas.sprites;

import java.util.ArrayList;
import java.lang.Runnable;

import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.entity.IEntity;
importorg.anddev.andengine.entity.modifier.MoveXModifier;
importorg.anddev.andengine.entity.modifier.MoveYModifier;
import org.anddev.andengine.entity.modifier.RotationModifier;
import org.anddev.andengine.entity.modifier.ScaleModifier;
import org.anddev.andengine.entity.scene.Scene;

import org.anddev.andengine.entity.sprite.Sprite;
importorg.anddev.andengine.entity.sprite.TiledSprite;
import org.anddev.andengine.entity.text.ChangeableText;
import org.anddev.andengine.input.touch.TouchEvent;
import org.anddev.andengine.opengl.font.Font;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
importorg.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.opengl.util.GLHelper;
importorg.anddev.andengine.opengl.vertex.RectangleVertexBuffer;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Baraja;
import xnetcom.pro.cartas.juego.Carta;
import xnetcom.pro.cartas.jugadores.JugadorHumano;

importandroid.util.Log;


publicclass MiSprite extends MiBaseSprite {
	// ===========================================================
	// Constants
	// ===========================================================

	// ===========================================================
	// Fields
	// ===========================================================

	// ===========================================================
	// Constructors
	// ===========================================================
	private SpriteDorso Dorso;
	private Scene scene;


	private ChangeableText nombre;
	private ChangeableText salud;
	private ChangeableText PIDH;
	private ChangeableText educacion;
	private ChangeableText pib;
	private ChangeableText continente;

	private ChangeableText Bsalud;
```

```
	private ChangeableText BPIDH;
	private ChangeableText Beducacion;
	private ChangeableText Bpib;
	privateintmovedores=0;



	private Font mFont;
	private Sprite logo;
	private Game juego;
	privateintnumero;
	privatebooleanbocaAbajo=false;
	privatebooleanRecienPuestoPocAbajo=false;
	privatebooleanRecienPuestoPocArriba=false;
	privateintseleccion=0;
	intoldX=0;

	intoldY=0;
	intmovidoX=4;
	intmovidoY=0;
	booleanmovido=false;
	booleanmovidoRapido=false;
	longtiempo;
	longtiempo2;
	ScaleModifier SM;
	MiSprite spr=this;
	booleanpillado=false;
	booleanpillado2=false;



	/*
	 * colocada
	 * 1 colocadaizquierda
	 * 2 colocadacolocadaderecha
	 */
	privateintcolocada=2;
	private Carta carta;
	privateintZindex;


	public MiSprite(Game juego,Carta carta,finalfloat pX, finalfloat pY,
TextureRegion pTextureRegion,TextureRegion bandera,TextureRegion dorso,Font
fuente,Font Bfuente,int numero) {

		super(pX, pY, pTextureRegion.getWidth(),
pTextureRegion.getHeight(), pTextureRegion);
		this.scene=juego.getScene();
		this.carta=carta;
		this.juego=juego;
		this.numero=numero;

		logo= new Sprite(0, 40, bandera);

		Dorso=new SpriteDorso(0, 0, dorso);
		this.Dorso.setRotation(180f);
		mFont=fuente;
```

```
            this.nombre= new ChangeableText(10, 3, this.mFont, " ", "nombre
del pais mas largo".length());

            nombre.setText(carta.getNombre());
            nombre.setPosition((this.getWidth()/2f)-nombre.getWidth()/2f,
5);

            this.PIDH= new ChangeableText(10, 176, this.mFont, " ", "nombre
del pais mas largo largo larggooo".length()){
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent, finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
                        //System.out.println("tocado textoPIDH");
                        setSeleccion(1);
                        super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX, pTouchAreaLocalY);
                        returnfalse;
                  }
            };
            PIDH.setText("IDH: "+carta.getIdh()+" - Nº:
"+Integer.toString(carta.getPidh()));

            this.BPIDH= new ChangeableText(10, 176, Bfuente, " ", "nombre
del pais mas largo largo larggooo".length());
            BPIDH.setText("IDH: "+carta.getIdh()+" - Nº:
"+Integer.toString(carta.getPidh()));
            BPIDH.setVisible(false);

            this.salud= new ChangeableText(10, 217, this.mFont, " ", "nombre
del pais mas largo".length()){
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent, finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
                        //System.out.println("tocadoSalud");
                        setSeleccion(2);
                        super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX, pTouchAreaLocalY);
                        returnfalse;
                  }
            };
            salud.setText("Salud: "+Float.toString(carta.getSalud()));

            this.Bsalud= new ChangeableText(10, 217, Bfuente, " ", "nombre
del pais mas largo".length());
            Bsalud.setText("Salud: "+Float.toString(carta.getSalud()));
            Bsalud.setVisible(false);

            this.educacion= new ChangeableText(10, 257, this.mFont, " ",
"nombre del pais mas largo".length()){
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent, finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
                        System.out.println("tocado texto Salud");
                        setSeleccion(3);
                        super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX, pTouchAreaLocalY);
                        returnfalse;
                  }
            };
            educacion.setText("Educacion:
"+Float.toString(carta.getEducacion()));
```

```
            this.Beducacion= new ChangeableText(10, 257, Bfuente, " ",
"nombre del pais mas largo".length());
            Beducacion.setText("Educacion:
"+Float.toString(carta.getEducacion()));
            Beducacion.setVisible(false);

            this.pib= new ChangeableText(10, 295, this.mFont, " ", "nombre
del pais mas largo".length()){
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent, finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
                        //System.out.println("tocadotextopib");
                        setSeleccion(4);
                        super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX, pTouchAreaLocalY);
                        returnfalse;
                  }
            };
            pib.setText("Ingresos: "+Float.toString(carta.getIngresos()));

            this.Bpib= new ChangeableText(10, 295, Bfuente, " ", "nombre del
pais mas largo".length()){
                  publicboolean onAreaTouched(final TouchEvent
pSceneTouchEvent, finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
                        //System.out.println("tocadotextopib");
                        setSeleccion(4);
                        super.onAreaTouched(pSceneTouchEvent,
pTouchAreaLocalX, pTouchAreaLocalY);
                        returnfalse;
                  }
            };
            Bpib.setText("Ingresos: "+Float.toString(carta.getIngresos()));
            Bpib.setVisible(false);

            this.continente= new ChangeableText(10, 292, this.mFont, " ",
"nombre del pais mas largo".length());
            continente.setText(carta.getContinente());
            continente.setPosition((this.getWidth()/2f)-
continente.getWidth()/2f, 340);
      }

      publicboolean click(){
            if(System.currentTimeMillis()-tiempo<=600&&!movido)returntrue;
            elsereturnfalse;
      }

      publicsynchronizedvoid moverINI(){

            movedores++;
            //Log.i(this.getNombre()+" movedores++","= "+movedores);
      }
      publicsynchronizedvoid MoverFIN(){
            movedores--;
            //Log.i(this.getNombre()+" movedores--","= "+movedores);
      }
      publicsynchronizedboolean enMarcha(){
            if(movedores==0)returnfalse;
            elsereturntrue;
      }
```

```
	publicboolean onAreaTouched(final TouchEvent pSceneTouchEvent,
	finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
		 System.out.println("tocado spritee");
		System.out.println(" carta.getSituacion()"+
carta.getSituacion());
		// this.setPosition(pSceneTouchEvent.getX() - this.getWidth() /
2,
		// pSceneTouchEvent.getY() - this.getHeight() / 2);

		// 0 apretar
		// 2 arrastrar
		// 1 soltar
		// Log.i("Card ZIndex", Integer.toString(this.getZIndex()));
		System.out.println("carta en index de humano
"+juego.getDirectorJuego().gethumano().getBaraja().getList().indexOf(carta));
		if (!juego.getDirectorTactil().pillar(numero)) {
			//System.out.println("soy
"+this.getNombre()+"syyatienenpilladpo");
			returnfalse;
		}

		//paraque no
sepuedantocarotrascartassitienesqueselecionarunacartaparajugar

	if(juego.getDirectorJuego().gethumano().CartaVista()&&(carta.getSituac
ion()!=4&&carta.getSituacion()!=5)){
			returnfalse;
		}
		if (carta.getSituacion()==0)returnfalse;


		// System.out.println("posicion="+getX());
		switch (pSceneTouchEvent.getAction()) {

		case 0:
			oldX = (int) pSceneTouchEvent.getX();
			oldY = (int) pSceneTouchEvent.getY();

			movidoX = 0;
			movidoY = 0;
			// unregisterEntityModifier(SM);
			// SM= new ScaleModifier(0.2f,getScaleX(),1);
			// registerEntityModifier(SM);
			// System.out.println("pillado"+carta);//
			tiempo = System.currentTimeMillis();
			pillado = true;
			//
System.out.println("pTouchAreaLocalX"+pTouchAreaLocalX);
			//
System.out.println("pTouchAreaLocalY"+pTouchAreaLocalY);

			break;

		case 1:
			movidoX = 0;
			movidoY = 0;
			// System.out.println("soltado"+carta);
			if (click()&&carta.getSituacion() == 1) {
				// System.out.println("tamano"+getScaleX());
```

```
                        if (getScaleX() >= 0.1f && getScaleX() < 0.3f){
                              //izq
                              if(getX()<100)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),10));

                              //centro

      if(getX()>100&&getX()<400)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),274));

                              //dere
                        if(getX()>500)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),532));
                        this.setZIndex(getZIndex()+1000);
                        scene.sortChildren();
                        agrandar(0.5f);

                        }

                        if (getScaleX() >= 0.9f && getScaleX() < 1.1f){
                              //izq
                              if(getX()<100)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),-50));

                              //centro

      if(getX()>100&&getX()<500)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),274));

                              //dere
                              if(getX()>500)this.registerEntityModifier(new
MoverX(0.5f,this.getX(),600));
                              this.setZIndex(getZIndex()-1000);
                              scene.sortChildren();
                              enpequenecer(0.5f);
                        }


                        /*
                         * if(getEjeZ()==-1500)acercar();
if(getEjeZ()==0)alejar();
                        */
                  }
                  // unregisterEntityModifier(SM);
                  // SM=new ScaleModifier(0.2f,getScaleX(),0.2f);



                  if (carta.getSituacion() ==
3||carta.getSituacion()==4||carta.getSituacion()==5) {
                        // cuandosueltasmira el angulo y dejalascartasensu
                        // posicion
                        long tiempoArrastre = System.currentTimeMillis() -
tiempo;
                        long distancia = 3 * Math.abs(oldX - (int)
pSceneTouchEvent.getX());
                        long velocidad = distancia / tiempoArrastre;
                        if (velocidad >= 1)
                              movidoRapido = true;
```

```
                        // SI LANZAS LA CARTA LLEGARA A SU SITIO SOLA
                        if (movido&&movidoRapido) {

                              //COLOCAR LA CARTA ANTES SELECCIONADA COMO
UNA MAS
                              if
(carta.getSituacion()==4||carta.getSituacion()==5||carta.getSituacion()==6){
                                    juego.getTic().esconder();
                                    try {
                                          scene.unregisterTouchArea(PIDH);

      scene.unregisterTouchArea(salud);

      scene.unregisterTouchArea(educacion);
                                          scene.unregisterTouchArea(pib);
                                    } catch (Exception e) {   }

                                    //registerEntityModifier(new
ScaleModifier(0.4f,this.getScaleX(), 1f));
                                    this.agrandar(0.4f);
                                    registerEntityModifier(new
MoverY(0.4f,getY(), getY() + 30));

      carta.getJugador().setCartaVista(false);
                              }

                              // movidohacialaizq
                              //System.out.println("oldX-getX() ="   +
(oldX - (int) pSceneTouchEvent.getX()));
                              if (oldX - (int) pSceneTouchEvent.getX() > 0)
{

                                    if
(carta.getSituacion()==4||carta.getSituacion()==5){
                                          setSeleccion(0);
                                          setZIndex(getZIndex()-2000);
                                          scene.sortChildren();
                                          //laescena a 3
akiparaqueseveacomodalavuelta
                                          carta.setSituacion(3);
                                    }

                                    registerEntityModifier(new
MoverX(0.4f, getX(),      91) {
                                          protectedvoid
onModifierFinished(IEntity pItem) {
                                                MiSprite df = (MiSprite)
pItem;

      df.setRecienPuestoBocaAbajo(true);

      super.onModifierFinished(pItem);
                                          }
                                    });
                              }

                              // movidohacialaderecha
```

```
                                if (oldX - (int) pSceneTouchEvent.getX() < 0)
{
                                      float velo=0.4f;
                                      if
(carta.getSituacion()==4||carta.getSituacion()==5){
                                            setSeleccion(0);
                                            velo=0.2f;
                                      }
                                      //sile echo paraladerecha y
yatengotodas a laderecharecojo el mazo
                                      if(getX()>=452){
                                            System.out.println("deberia
regore");
                                            Baraja baraja
=juego.getDirectorJuego().gethumano().getBaraja();
                                            ArrayList<Carta>
list=baraja.getList();
                                            if(list.get(list.size()-
1).getSprite().equals(this)){

      juego.getDirector3D().recogerMazo();
                                            }
                                      }
                                      registerEntityModifier(new
MoverX(velo, getX(),452) {
                                            protectedvoid
onModifierFinished(IEntity pItem) {
                                                  MiSprite df = (MiSprite)
pItem;
                                                  if
(carta.getSituacion()==4||carta.getSituacion()==5){
                                                        //laescena a 3 aki
para que se no se de la vuelta la vuelta

      carta.setSituacion(3);

      setZIndex(getZIndex()-2000);

      scene.sortChildren();
                                                  }


      df.setRecienPuestoBocaArriba(true);

      super.onModifierFinished(pItem);
                                            }
                                      });
                                }
                          }
                          //la ponemor para coger su seleccion
                          if (click()&&!getBocaAbajo()&&carta.getSituacion()
== 3&&!carta.getJugador().CartaVista()){
                                scene.registerTouchArea(PIDH);
                                scene.registerTouchArea(salud);
                                scene.registerTouchArea(educacion);
                                scene.registerTouchArea(pib);
```

```
      if(carta.getJugador().getTipojugador()==1&&carta.getJugador().turno())
{
                                    //tenemos que elegir un valor
                                    if (carta.getJugador().sacarCarta()) {

                                          carta.setSituacion(4);
                                          //no tenemos que elegir un valor
solo la carta
                                    }else{
                                          carta.setSituacion(5);
                                          JugadorHumano jug=
(JugadorHumano)carta.getJugador();


      carta.setSeleccion(jug.getSel());
                                          juego.getTic().mostrar(carta);
                                    }
                                    this.setZIndex(this.getZIndex()+2000);
                                    scene.sortChildren();

      carta.getJugador().setCartaVista(true);
                                    this.setRotation(0);
                                    registerEntityModifier(new
ScaleModifier(0.4f,this.getScaleX(), 1.2f));
                                    registerEntityModifier(new
MoverX(0.4f,getX(), Game.CAMERA_WIDTH/2-this.getWidthScaled()/2));
                                    registerEntityModifier(new
MoverY(0.4f,getY(), getY() - 30));
                              }

                        }

                  }

                  // PARTE QUE LE PERMMITE SOLTAR LA CARTA Y QUE SE COLOQUE
SOLA EN LA
                  // PARTE CORRECTA
                  if (movido&& !movidoRapido&&carta.getSituacion() == 3) {
                        if ((getRotation() <= -90f)) {

                              registerEntityModifier(new MoverX(0.4f,
getX(), 91) {
                                    protectedvoid
onModifierFinished(IEntity pItem) {
                                          MiSprite df = (MiSprite) pItem;

      df.setRecienPuestoBocaAbajo(true);
                                          super.onModifierFinished(pItem);
                                    }
                              });

                        } else {
                              registerEntityModifier(new MoverX(0.4f,
getX(), 452) {
                                    protectedvoid
onModifierFinished(IEntity pItem) {
                                          MiSprite df = (MiSprite) pItem;
```

```
		df.setRecienPuestoBocaArriba(true);
                                                super.onModifierFinished(pItem);
                                        }
                                });
                        }
                }

                // registerEntityModifier(SM);
                movido = false;
                pillado = false;
                juego.getDirectorTactil().soltar();
                movidoRapido = false;
                break;

        case 2:
                if (carta.getSituacion() == 1)break;
                if (pillado) {
                        //System.out.println("posicion=" + this.getX()+ " Y
LA ROTACION =" + this.getRotation());

                        movido = true;
                        // setPosition(getX()+pSceneTouchEvent.getX()-oldX-
movidoX,
                        // getY()+pSceneTouchEvent.getY()-oldY-movidoY);

		if(carta.getSituacion()!=4&&carta.getSituacion()!=5)setPosition(getX()
+ pSceneTouchEvent.getX() - oldX - movidoX, getY());
                        //System.out.println("arrastrado"+getZIndex());
                        movidoX = (int) (pSceneTouchEvent.getX() - oldX);
                        movidoY = (int) (pSceneTouchEvent.getY() - oldY);
                }

                break;
        }
        returntrue;
	}
	publicvoid recoloca(){
        if (carta.getSituacion() == 1) {

                if (getScaleX() >= 0.9f && getScaleX() < 1.1f) {
                        // izq
                        if (getX() < 100)
                                this.registerEntityModifier(new MoverX(0.5f,
this.getX(),
                                                -50));

                        // centro
                        if (getX() > 100 && getX() < 500)
                                this.registerEntityModifier(new MoverX(0.5f,
this.getX(),
                                                274));

                        // dere
                        if (getX() > 500)
                                this.registerEntityModifier(new MoverX(0.5f,
this.getX(),
                                                600));
                        this.setZIndex(getZIndex() - 1000);
```

```
                        scene.sortChildren();
                        enpequenecer(0.5f);
                  }
            }

      }



      publicboolean peque(){
            if(getScaleX()>=0.1f&&getScaleX()<0.3f) returntrue;
            elsereturnfalse;
      }
      publicvoid verDorso(){
            this.EjeaRotar(0, 1, 0);
            this.setRotation(-180f);
      }
      publicvoid verCarta(){
            this.EjeaRotar(0, 1, 0);
            this.setRotation(0f);
      }
      publicvoid verCarta(float duracion){
            this.EjeaRotar(0, 1, 0);
            this.registerEntityModifier(new
RotationModifier(duracion,this.getRotation(),0));
      }
      publicvoid verDorso(float duracion){
            this.EjeaRotar(0, 1, 0);
            this.registerEntityModifier(new
RotationModifier(duracion,this.getRotation(),-180));
      }

      publicvoid agrandar(float duracion){
            this.registerEntityModifier(new
ScaleModifier(duracion,this.getScaleX(),1f));
      }
      publicvoid enpequenecer(float duracion){
            this.registerEntityModifier(new
ScaleModifier(duracion,this.getScaleX(),0.2f));
      }
      publicsynchronizedvoid setRecienPuestoBocaAbajo(boolean resultado){
            RecienPuestoPocAbajo=resultado;
      }

      publicsynchronizedvoid setRecienPuestoBocaArriba(boolean resultado){
            RecienPuestoPocArriba=resultado;
      }

      publicsynchronizedboolean getRecienPuestoBocaArriba(){
            returnRecienPuestoPocArriba;
      }

      publicsynchronizedboolean getRecienPuestoBocaAbajo(){
            returnRecienPuestoPocAbajo;
      }

      publicfloat getPosisionCentro(){
            return (this.getX()+ this.getWidth()/2);
      }
```

```
publicvoid setSeleccion(int seleccion){
	if(this.seleccion==seleccion) return;
	if(carta.getSituacion()!=4) return;
	if(seleccion!=0)juego.getTic().mostrar(carta);
	switch (seleccion) {

	case 0:
		carta.setSeleccion(0);
		PIDH.setVisible(true);
		BPIDH.setVisible(false);
		salud.setVisible(true);
		Bsalud.setVisible(false);
		educacion.setVisible(true);
		Beducacion.setVisible(false);
		pib.setVisible(true);
		Bpib.setVisible(false);
		break;
	case 1:
		carta.setSeleccion(1);
		PIDH.setVisible(false);
		BPIDH.setVisible(true);
		salud.setVisible(true);
		Bsalud.setVisible(false);
		educacion.setVisible(true);
		Beducacion.setVisible(false);
		pib.setVisible(true);
		Bpib.setVisible(false);
		break;
	case 2:
		carta.setSeleccion(2);
		PIDH.setVisible(true);
		BPIDH.setVisible(false);
		salud.setVisible(false);
		Bsalud.setVisible(true);
		educacion.setVisible(true);
		Beducacion.setVisible(false);
		pib.setVisible(true);
		Bpib.setVisible(false);
		break;
	case 3:
		carta.setSeleccion(3);
		PIDH.setVisible(true);
		BPIDH.setVisible(false);
		salud.setVisible(true);
		Bsalud.setVisible(false);
		educacion.setVisible(false);
		Beducacion.setVisible(true);
		pib.setVisible(true);
		Bpib.setVisible(false);
		break;
	case 4:
		carta.setSeleccion(4);
		PIDH.setVisible(true);
		BPIDH.setVisible(false);
		salud.setVisible(true);
		Bsalud.setVisible(false);
		educacion.setVisible(true);
		Beducacion.setVisible(false);
```

```
                    pib.setVisible(false);
                    Bpib.setVisible(true);
                    break;

             }

             this.seleccion=seleccion;
      }


      //es la unica forma segura de borrar las cartas
publicvoid remover(MiSprite spr){
      final MiSprite sprt=spr;
      juego.runOnUpdateThread(new Runnable() {
@Override
publicvoid run() {
/* Now it is save to remove the entity! */
scene.detachChild(sprt);
         }
 });


}


      publicvoid setSeleccionBot(int seleccion){
             switch (seleccion) {

             case 0:
                    carta.setSeleccion(0);
                    PIDH.setVisible(true);
                    BPIDH.setVisible(false);
                    salud.setVisible(true);
                    Bsalud.setVisible(false);
                    educacion.setVisible(true);
                    Beducacion.setVisible(false);
                    pib.setVisible(true);
                    Bpib.setVisible(false);
                    break;
             case 1:
                    carta.setSeleccion(1);
                    PIDH.setVisible(false);
                    BPIDH.setVisible(true);
                    salud.setVisible(true);
                    Bsalud.setVisible(false);
                    educacion.setVisible(true);
                    Beducacion.setVisible(false);
                    pib.setVisible(true);
                    Bpib.setVisible(false);
                    break;
             case 2:
                    carta.setSeleccion(2);
                    PIDH.setVisible(true);
                    BPIDH.setVisible(false);
                    salud.setVisible(false);
                    Bsalud.setVisible(true);
                    educacion.setVisible(true);
                    Beducacion.setVisible(false);
                    pib.setVisible(true);
```

```
                Bpib.setVisible(false);
                break;
            case 3:
                carta.setSeleccion(3);
                PIDH.setVisible(true);
                BPIDH.setVisible(false);
                salud.setVisible(true);
                Bsalud.setVisible(false);
                educacion.setVisible(false);
                Beducacion.setVisible(true);
                pib.setVisible(true);
                Bpib.setVisible(false);
                break;
            case 4:
                carta.setSeleccion(4);
                PIDH.setVisible(true);
                BPIDH.setVisible(false);
                salud.setVisible(true);
                Bsalud.setVisible(false);
                educacion.setVisible(true);
                Beducacion.setVisible(false);
                pib.setVisible(false);
                Bpib.setVisible(true);
                break;

            }

            this.seleccion=seleccion;



      }
      protectedvoid onManagedUpdate(float pSecondsElapsed) {
            //para optimizar l rendimiento se hacen aparecer y desaparecer
las cartas del usuarion jhumano
            if(carta.getSituacion()==3){
                if(getBocaAbajo()&&getRecienPuestoBocaAbajo()){

                    int pos
=juego.getDirectorJuego().gethumano().getBaraja().getPosicion(carta);
                    //System.out.println("RecienPuestoPocAbajo y su
posicion es "+pos+ "va,mos a intertar hacer invisible "+(pos-2)+" y hacer
bisible "+(pos +2) );
                    try{

      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos+3).
getSprite().setVisible(false);

                    }catch(Exception e){}

                    try{

      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
1).getSprite().setVisible(true);

      juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
2).getSprite().setVisible(true);
```

```
	juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
3).getSprite().setVisible(true);
				}catch(Exception e){}
				setRecienPuestoBocaAbajo(false);
			}

			if(!getBocaAbajo()&&getRecienPuestoBocaArriba()){

				int pos
=juego.getDirectorJuego().gethumano().getBaraja().getPosicion(carta);
				//System.out.println("RecienPuestoPocArriba y su
posicion es "+pos+ "va,mos a intertar hacer invisible "+(pos+2)+" y hacer
bisible "+(pos-2) );
				try{

	juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos+1).
getSprite().setVisible(true);

	juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos+2).
getSprite().setVisible(true);

	juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos+3).
getSprite().setVisible(true);
				}catch(Exception e){}

				try{

	juego.getDirectorJuego().gethumano().getBaraja().getList().get(pos-
3).getSprite().setVisible(false);
				}catch(Exception e){}
				setRecienPuestoBocaArriba(false);
			}



			miraracartas();
		}
super.onManagedUpdate(pSecondsElapsed);

	}

	protectedvoid miraracartas(){

		if (carta.getSituacion() == 3) {

			if (getX() <= 91) {
				EjeaRotar(0, 1, 0);
				setRotation(-180);
			}

			if ((Game.CAMERA_WIDTH / 2) - getX() <= 0) {
				EjeaRotar(0, 1, 0);
				setRotation(0);
			}

			if (((400 + 180) - getPosisionCentro()) > 0 && getX() >
91) {
				EjeaRotar(0, 1, 0);
```

```
                        this.setRotation(-(400 + 180 - getPosisionCentro())
/ 2);
                  }

                  // Log.i("Card ZIndex="+this.getNombre(),
                  // Integer.toString(this.getZIndex()));
                  // System.out.println(getNombre()+" Z"+getZIndex());
                  if (getRotation() < -90f) {
                        if (!getBocaAbajo()) {
                              setBocaAbajo(true);
                              //setParent(scene);
                              setZIndex(2000 - getZIndex());
                              scene.sortChildren();

                              // ArrayList<ITouchArea> ita
=scene.getTouchAreas();
                              // scene.clearTouchAreas();
                              // ita.iterator()

                              System.out.println("getRotation()<=-90f
cambiamos el z "+ getZIndex());
                        }
                  }
                  if (getRotation() > -90f) {
                        if (getBocaAbajo()) {
                              setBocaAbajo(false);
                              //setParent(scene);
                              setZIndex(2000 - getZIndex());
                              scene.sortChildren();
                        }
                  }
            }
      }

      publicvoid pais (){
            //logo.setCurrentTileIndex(1, 1);
      }

      publicvoid setIDH(String idh){
            PIDH.setText(Integer.toString(carta.getPidh()));

      }


      publicvoid setBocaAbajo(boolean bocaAbajo){
            this.bocaAbajo=bocaAbajo;
      }
      publicboolean getBocaAbajo(){
            returnbocaAbajo;
      }
      publicvoid registrar(){
            scene.attachChild(this);
            this.attachChild(PIDH);
            this.attachChild(nombre);
            this.attachChild(salud);
            this.attachChild(educacion);
            this.attachChild(pib);
            this.attachChild(BPIDH);
```

```
            this.attachChild(Bsalud);
            this.attachChild(Beducacion);
            this.attachChild(Bpib);
            this.attachChild(continente);
            this.attachChild(logo);
            this.attachChild(Dorso);
            //scene.registerTouchArea(logo);
            //scene.registerTouchArea(PIDH);
            //scene.registerTouchArea(salud);
            //scene.registerTouchArea(educacion);
            //scene.registerTouchArea(pib);

      }
      // ===========================================================
      // Getter & Setter
      // ===========================================================

      // ===========================================================
      // Methods for/from SuperClass/Interfaces
      // ===========================================================

      @Override
      public TextureRegion getTextureRegion() {
            return (TextureRegion)this.mTextureRegion;
      }

      publicvoid setTextureRegion(TextureRegion pTextureRegion) {
            this.mTextureRegion=pTextureRegion;
      }


      @Override// esta funcion modificada ahora me permite mandarle a opengl
que traslade los sprites mas lejos
      protectedvoid applyTranslation(final GL10 pGL) {
            pGL.glTranslatef(this.mX, this.mY, this.mZ);
      }



      publicvoid acercar(){
            if(getEjeZ()!=-1500)return;
            registerEntityModifier(new MoveZModifier(2f,-1500,0){
                  @Override
                  protectedvoid onModifierStarted(
                              IEntity pItem) {
                        // TODO Auto-generated method stub
                        super.onModifierStarted(pItem);
                  }

                  @Override
                  protectedvoid onModifierFinished(
                              IEntity pItem) {

                        // TODO Auto-generated method stub
                        super.onModifierFinished(pItem);
                  }
            });
```

```
	}
	publicvoid alejar(){
		if(getEjeZ()!=0)return;
		registerEntityModifier(new MoveZModifier(2f,0,-1500){
			@Override
			protectedvoid onModifierStarted(
					IEntity pItem) {
				// TODO Auto-generated method stub
				super.onModifierStarted(pItem);
			}

			@Override
			protectedvoid onModifierFinished(
					IEntity pItem) {
				// TODO Auto-generated method stub
				super.onModifierFinished(pItem);
			}
		});
	}


	public SpriteDorso getDorso(){

		returnDorso;
	}

	publicvoid EjeaRotar(int x,int y, int z){
		this.rX=x;
		this.rY=y;
		this.rZ=z;
		//IDH.EjeaRotar(x, y, z);
		//Dorso.EjeaRotar(x, y, z);
	}


	@Override
	protectedvoid applyRotation(final GL10 pGL) {
		GLHelper.enableCulling(pGL);
		finalfloat rotation = this.mRotation;

		if(rotation != 0) {

			finalfloat rotationCenterX = this.mRotationCenterX;
			finalfloat rotationCenterY = this.mRotationCenterY;
			pGL.glTranslatef(rotationCenterX, rotationCenterY, 0);
			//* Note we are applying rotation around the y-axis and
not the z-axis anymore!
			pGL.glRotatef(rotation, rX, rY, rZ);
			pGL.glTranslatef(-rotationCenterX, -rotationCenterY, 0);
			//this.getDorso().setRotation(rotation+180);
			//this.IDH.setRotation(rotation);
		}
	}



	@Override
	publicvoid setZIndex(int pZIndex) {
		// TODO Auto-generated method stub
```

```
		//IDH.setZIndex(pZIndex);
		//Dorso.setZIndex(pZIndex);

		super.setZIndex(pZIndex);
	}
	publicint getIndex(){
		returnZindex;
	}
	publicvoid setIndex(int index){
		Zindex=index;
	}

	publicvoid censura() {
		BPIDH.setText("IDH: ---" + " - Nº: ---");
		PIDH.setText("IDH: ---" + " - Nº: ---");
		salud.setText("Salud: ----");
		Bsalud.setText("Salud: ----");
		educacion.setText("Educacion: ----");
		Beducacion.setText("Educacion: ----");
		BPIDH.setText("IDH: ----");
		pib.setText("Ingresos: ----");
		Bpib.setText("Ingresos: ----");
	}

	public String getNombre(){

		returnthis.carta.getNombre();
	}

	public Carta getCarta(){
		returncarta;
	}

	// funcion que mejora el renderizado para las cartas
	@Override
	protectedvoid drawVertices(final GL10 pGL, final Camera pCamera) {
		pGL.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_NICEST);
		super.drawVertices(pGL, pCamera);


	}

}
```

Archivo MoverX.java

```
package xnetcom.pro.cartas.sprites;

import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.modifier.SingleValueSpanEntityModifier;



publicclass MoverX extends SingleValueSpanEntityModifier{

	public MoverX(float pDuration, float pFromValue, float pToValue) {
		super(pDuration, pFromValue, pToValue);
		// TODO Auto-generated constructor stub
	}

	protected MoverX(final MoverX moveXModifier) {
		super(moveXModifier);
	}



	public MoverX deepCopy(){
		returnnew MoverX(this);
	}

	@Override
	protectedvoid onModifierStarted(IEntity pItem) {
		MiSprite df;
		df=(MiSprite) pItem;
		df.moverINI();
		// TODO Auto-generated method stub
		super.onModifierStarted(pItem);
	}

	@Override
	protectedvoid onModifierFinished(IEntity pItem) {
		MiSprite df;
		df=(MiSprite) pItem;
		df.MoverFIN();
		// TODO Auto-generated method stub
		super.onModifierFinished(pItem);
	}

	protectedvoid onSetInitialValue(final IEntity pEntity, finalfloat pX)
{
		pEntity.setPosition(pX, pEntity.getY());
	}


	protectedvoid onSetValue(final IEntity pEntity, finalfloat
pPercentageDone, finalfloat pX) {
		pEntity.setPosition(pX, pEntity.getY());
	}


}
```

Archivo MoverY.java

```
package xnetcom.pro.cartas.sprites;

import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.modifier.SingleValueSpanEntityModifier;



publicclass MoverY extends SingleValueSpanEntityModifier{

	public MoverY(float pDuration, float pFromValue, float pToValue) {
		super(pDuration, pFromValue, pToValue);
		// TODO Auto-generated constructor stub
	}

	protected MoverY(final MoverY moveYModifier) {
		super(moveYModifier);
	}



	public MoverY deepCopy(){
		returnnew MoverY(this);
	}

	@Override
	protectedvoid onModifierStarted(IEntity pItem) {
		MiSprite df;
		df=(MiSprite) pItem;
		df.moverINI();
		// TODO Auto-generated method stub
		super.onModifierStarted(pItem);
	}

	@Override
	protectedvoid onModifierFinished(IEntity pItem) {
		MiSprite df;
		df=(MiSprite) pItem;
		df.MoverFIN();
		// TODO Auto-generated method stub
		super.onModifierFinished(pItem);
	}

	protectedvoid onSetInitialValue(final IEntity pEntity, finalfloat pY)
{
		pEntity.setPosition(pEntity.getX(),pY);
	}


	protectedvoid onSetValue(final IEntity pEntity, finalfloat
pPercentageDone, finalfloat pY) {
		pEntity.setPosition( pEntity.getX(),pY);
	}


}
```

Archivo MoverZModifier.java

```
package xnetcom.pro.cartas.sprites;

import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.modifier.SingleValueSpanEntityModifier;
import org.anddev.andengine.util.modifier.ease.IEaseFunction;



publicclass MoveZModifier extends SingleValueSpanEntityModifier {
	public MoveZModifier(finalfloat pDuration, finalfloat pFromZ,
finalfloat pToZ) {
		this(pDuration, pFromZ, pToZ, null, IEaseFunction.DEFAULT);
	}

	public MoveZModifier(finalfloat pDuration, finalfloat pFromZ,
finalfloat pToZ, final IEaseFunction pEaseFunction) {
		this(pDuration, pFromZ, pToZ, null, pEaseFunction);
	}

	public MoveZModifier(finalfloat pDuration, finalfloat pFromZ,
finalfloat pToZ, final IEntityModifierListener pEntityModifierListener) {
		super(pDuration, pFromZ, pToZ, pEntityModifierListener,
IEaseFunction.DEFAULT);
	}

	public MoveZModifier(finalfloat pDuration, finalfloat pFromZ,
finalfloat pToZ, final IEntityModifierListener pEntityModifierListener, final
IEaseFunction pEaseFunction) {
		super(pDuration, pFromZ, pToZ, pEntityModifierListener,
pEaseFunction);
	}

	protected MoveZModifier(final MoveZModifier moveZModifier) {
		super(moveZModifier);
	}


	public MoveZModifier deepCopy(){
		returnnew MoveZModifier(this);
	}

	protectedvoid onSetInitialValue(final IEntity pEntity, finalfloat pZ)
{
		MiSprite df;
		df=(MiSprite) pEntity;
		df.translationZ(pZ);
	}


	protectedvoid onSetValue(final IEntity pEntity, finalfloat
pPercentageDone, finalfloat pZ) {
		MiSprite df;
		df=(MiSprite) pEntity;
		df.translationZ(pZ);
	}
}
```

Archivo SpriteBoton.java

```
package xnetcom.pro.cartas.sprites;


import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.input.touch.TouchEvent;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import xnetcom.pro.cartas.activities.MainMenu;

publicclass SpriteBoton extends AnimatedSprite{

	private MainMenu menu;
	public SpriteBoton(MainMenu menu, float pX, float
pY,TiledTextureRegion pTiledTextureRegion) {
		super(pX, pY, pTiledTextureRegion);
		this.menu=menu;
		this.setScaleX(1.3f);
		// TODO Auto-generated constructor stub
	}

	publicboolean onAreaTouched(final TouchEvent pSceneTouchEvent,
	finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {


		switch(pSceneTouchEvent.getAction()){
		case 0:
			//0 apretar
			setCurrentTileIndex(1);
		break;
		case 1:
			//1 soltar
			setCurrentTileIndex(0);
			accion();
		break;

		}
		returntrue;
	}

	publicvoid accion(){
		//menu.cagarBaraja();
	}


}
```

Archivo SpriteDorso.java

```
package xnetcom.pro.cartas.sprites;


import javax.microedition.khronos.opengles.GL10;

import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.util.GLHelper;
importorg.anddev.andengine.opengl.vertex.RectangleVertexBuffer;


publicclass SpriteDorso extends MiBaseSprite{




	public SpriteDorso(finalfloat pX, finalfloat pY,TextureRegion
pTextureRegion) {
		super(pX, pY, pTextureRegion.getWidth(),
pTextureRegion.getHeight(), pTextureRegion);
	}

	public SpriteDorso(finalfloat pX, finalfloat pY, finalfloat pWidth,
finalfloat pHeight, final TextureRegion pTextureRegion) {
		super(pX, pY, pWidth, pHeight, pTextureRegion);
	}


	public TextureRegion getTextureRegion() {
		return (TextureRegion)this.mTextureRegion;
	}

	publicvoid setTextureRegion(TextureRegion pTextureRegion) {
		this.mTextureRegion=pTextureRegion;
	}



	@Override
	publicvoid setRotation(float pRotation) {
		// TODO Auto-generated method stub
		super.setRotation(pRotation);
	}


	@Override
	publicvoid setRotationCenterX(float pRotationCenterX) {
		// TODO Auto-generated method stub
		super.setRotationCenterX(pRotationCenterX);
	}

	@Override
	publicvoid setRotationCenterY(float pRotationCenterY) {
		// TODO Auto-generated method stub
		super.setRotationCenterY(pRotationCenterY);
	}
```

```
	@Override
	publicvoid setRotationCenter(float pRotationCenterX, float
pRotationCenterY) {
		// TODO Auto-generated method stub
		super.setRotationCenter(pRotationCenterX, pRotationCenterY);
	}

	protectedvoid applyRotation(final GL10 pGL) {
		//* Disable culling so we can see the backside of this sprite.
		//GLHelper.disableCulling(pGL);
		GLHelper.enableCulling(pGL);
		finalfloat rotation = this.mRotation;

		if(rotation != 0) {
			finalfloat rotationCenterX = this.mRotationCenterX;
			finalfloat rotationCenterY = this.mRotationCenterY;

			pGL.glTranslatef(rotationCenterX, rotationCenterY, 0);
			//* Note we are applying rotation around the y-axis and
not the z-axis anymore!
			pGL.glRotatef(rotation, 0, 1, 0);
			pGL.glTranslatef(-rotationCenterX, -rotationCenterY, 0);
		}
	}

	publicvoid translationZ(float z) {
		super.translationZ(z);
	}
	protectedvoid applyTranslation(final GL10 pGL) {
		pGL.glTranslatef(this.mX, this.mY, this.mZ);
	}

	protectedvoid drawVertices(final GL10 pGL, final Camera pCamera) {
		pGL.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_NICEST);
		super.drawVertices(pGL, pCamera);

		///* Enable culling as 'normal' entities profit from culling.
		//GLHelper.enableCulling(pGL);
	}

}
```

Archivo SpriteTic.java

```
package xnetcom.pro.cartas.sprites;

import org.anddev.andengine.entity.IEntity;
import org.anddev.andengine.entity.modifier.ScaleModifier;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
importorg.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.input.touch.TouchEvent;
importorg.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;

import xnetcom.pro.cartas.activities.Game;
import xnetcom.pro.cartas.juego.Carta;



publicclass TicSprite extends AnimatedSprite{

	Game game;
	Carta carta;
	public TicSprite(Game game,float pX, float pY, TiledTextureRegion
pTiledTextureRegion) {
		super(pX, pY, pTiledTextureRegion);
		this.game=game;
		setZIndex(3000);
		this.setVisible(false);
		setCurrentTileIndex(0);
	}

	publicboolean onAreaTouched(final TouchEvent pSceneTouchEvent,
	finalfloat pTouchAreaLocalX, finalfloat pTouchAreaLocalY) {
		System.out.println("tocado tic");
		if(pSceneTouchEvent.getAction()==1){
			System.out.println("tocado tic setCurrentTileIndex(1)");
			setCurrentTileIndex(1);

		}

		if(pSceneTouchEvent.getAction()==0){
		//this.setVisible(false);
			System.out.println("tocado tic setCurrentTileIndex(0)");
			apretado();
		}
		returntrue;
	}


	publicvoid apretado(){
		game.getDirector3D().recogerMazo();
		//este scale essimplementeunretardo
		registerEntityModifier(new ScaleModifier(0.7f,this.getScaleX(),
this.getScaleX()){
			protectedvoid onModifierFinished(IEntity pItem) {

				esconder();

	game.getDirectorJuego().gethumano().cartaSeleccionada(carta);
				super.onModifierFinished(pItem);
```

```
                }

            });
        }

        publicvoid mostrar(Carta carta){
            this.carta=carta;
            setCurrentTileIndex(0);
            setVisible(true);
        }
        publicvoid esconder(){
            setVisible(false);
            setCurrentTileIndex(0);
        }

}
```